Secure Outsourcing of IT Services in a Non-Trusted Environment

DISSERTATION

zur Erlangung des akademischen Grades
doctor rerum politicarum
(Dr. rer. pol.)
im Fach Wirtschaftswissenschaft

eingereicht an der
Wirtschaftswissenschaftlichen Fakultät
Humboldt-Universität zu Berlin

von
Herr Dipl.-Math. Sergei Evdokimov
geboren am 15.03.1980 in Minsk

Präsident der Humboldt-Universität zu Berlin:
Prof. Dr. Dr. h.c. Christoph Markschies

Dekan der Wirtschaftswissenschaftlichen Fakultät:
Prof. Oliver Günther, Ph.D.

Gutachter:

1. Prof. Oliver Günther, Ph.D.
2. Prof. Bharat Bhargava, Ph.D.

Tag des Kolloquiums: 5. August 2008

Abstract

This thesis considers the possibilities of secure outsourcing of databases and of content-based routing operations to an untrusted service provider. We explore the limits of the security that is achievable in these scenarios. When discussing security, we refer to the state of the art definitions from cryptography and complexity theory. The key contributions of the thesis are the following:

- We explore the applicability of cryptographic constructs that allow performing operations over encrypted data, also known as privacy homomorphisms, for creating protocols that could enable secure database outsourcing. We also describe a framework for secure database outsourcing that is based on searchable encryption schemes, and prove its correctness and security.

- We describe a new searchable encryption scheme that exceeds existing analogues with regard to certain parameters: compared to the existing works, the proposed scheme allows for performing a larger number of operations over a securely outsourced database and has significantly lower chances of returning erroneous results of a search.

- We propose an approach for managing discretionary access to securely outsourced and encrypted databases. Compared to existing techniques, our approach is applicable to more general scenarios, is simpler and has similar performance characteristics.

- We examine possibilities of performing a secure content-based routing by building a formal security model that describes a secure content-based routing system, evaluate existing approaches against this model, and provide an analysis of the possibilities for achieving confidentiality when performing the routing. Compared to the existing works, which fail in providing complete confidentiality, our security model considers shortcomings of these solutions. We also describe a content-based routing system that satisfies this model and to the best of our knowledge is the first of its kind to provide a complete confidentiality.

Keywords:
Privacy, Security, Confidentiality, Access Control, Outsourcing, Database, Content-Based Routing

Zusammenfassung

In dieser Arbeit werden die Möglichkeiten sicherer Ausgliederung von Datenbanken und inhaltsbasiertem Routing an einen nicht voll vertrauenswürdigen Dienstanbieter betrachtet. Wir untersuchen die Grenzen der Sicherheit, die in diesem Szenario erreicht werden können. Sicherheit wird dabei unter Zuhilfenahme aktueller komplexitätstheoretischer Arbeiten definiert. Dies ermöglicht die Verwendung formaler Methoden zur Untersuchung der Bedingungen, unter denen verschiedene Grade von Sicherheit möglich sind. Die Beiträge dieser Dissertation sind im Einzelnen:

- Wir untersuchen die Eignung sog. Privacy-Homomorphismen, welche die Ausführung von Operationen auf verschlüsselten Daten erlauben. Dies dient der Entwicklung von Protokollen zur sicheren Datenbankausgliederung. Weiter beschreiben wir ein allgemeines Framework für sichere Datenbankausgliederung, das auf sog. Volltextsuch-Verschlüsselungsverfahren basiert. Ferner stellen wir einen Beweis für die Sicherheit und Korrektheit vor.

- Wir beschreiben ein neues Volltextsuch-Verschlüsselungsverfahren, das im Vergleich zu bekannten Arbeiten eine größere Anzahl verschiedener Operationen für das Datenbank-Outsourcing-Problem ermöglicht und signifikant niedrigere Fehlerraten hat.

- Wir schlagen einen Ansatz vor, um im Kontext der sicheren Datenbank-Auslagerung Blanko-Zugriffe auf die verschlüsselten Daten zu verwalten. Verglichen mit existierenden Techniken ist unser Ansatz anwendbar auf generellere Szenarien, ist einfacher und hat ähnliche Effizienzeigenschaften.

- Wir untersuchen die Möglichkeit des sicheren inhaltsbasierten Routings, in dem wir ein formales Sicherheitsmodell konstruieren, existierende Ansätze in diesem Modell bewerten und eine formale Analyse der Möglichkeit von Vertraulichkeit durchführen. Unser Sicherheitsmodell deckt die Unzulänglichkeiten der bestehenden Ansätze auf. Schließlich beschreiben wir ein inhaltsbasiertes Routingverfahren, welches das Modell erfüllt.

Schlagwörter:
Datenschutz, Sicherheit, Datenvertraulichkeit, Vertraulichkeit,
Zugriffskontrolle, Ausgliederung, Datenbanken, Inhaltsbasiertes Routing

Contents

List of Figures

Chapter 1

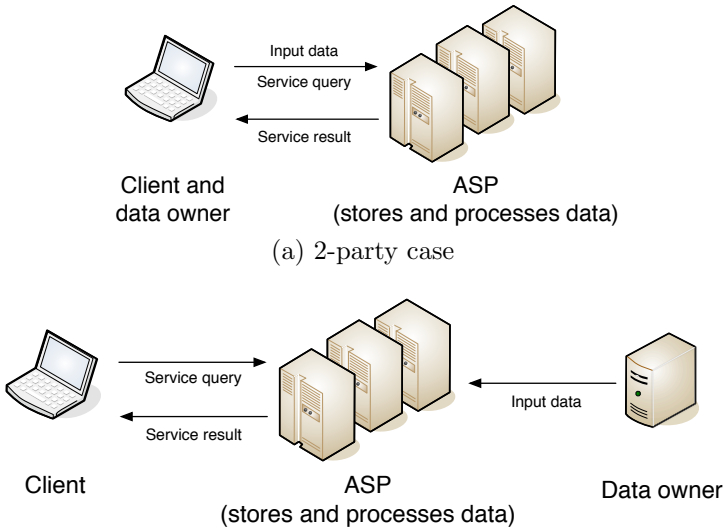
Introduction

In this chapter, we present a general introduction to the problem of secure data outsourcing, list our main results and outline the structure of the thesis.

1.1 Secure Outsourcing of IT Services

An Application Service Provider (ASP) is a business model that assumes that a company provides IT-services to its customers over a network. Examples of often outsourced services include web hosting, file hosting, application hosting, email services, CPU utilization for resource-intensive computations, etc. By serving multiple customers and benefiting from economies of scale, the ASP can provide the service at a lower cost in contrast to when service related operations were performed by the customers themselves.

Generally, an ASP architecture can be described as either 2-*party* or 3-*party* [Boy04] - Figures 1.1a - 1.1b. In the 2-party case, a service provider allows clients to use its advanced IT infrastructure and expertise for storing and processing their data. In the 3-party case, the data is used and provided by different parties and the service provider plays a role of the mediator that enables accumulation and processing of the data, and delivers them to the clients. Examples of the 2-party service outsourcing architecture are data mining service providers that perform data mining of the client's data, database service providers that provide clients with the possibility to store and process their data on remote databases, and computational centers that allow clients to perform resource-intensive computations on their hardware. To the 3-party case service outsourcing architecture, one could refer census bureaus that collect data from various agencies and provide it to the public, health initiatives that collect and analyze chronic disease data from hospitals and release it to researchers and patients, content-based routing systems



(a) 2-party case

(b) 3-party case

Figure 1.1: Possible ASP architectures

where a routing network is responsible for the delivery of data generated by content providers to subscribers interested in certain content.

The data processed by the ASP might be of arbitrary nature (medical, financial, personal, etc.), which in some cases cannot but raise privacy concerns from the side of the data owner. The main privacy risks of such scenarios are:

- ASP may not be trusted enough to be allowed to access the data
- data should not be accessible by unauthorized parties
- service queries may reveal private information about clients of the service
- ASP might (accidentally or deliberately) modify the data

All the listed risks arise only when the ASP is not trusted. And though "non malicious" and trusted behavior of the ASP can be enforced by a contract, in many cases their legal power might be not sufficient. Differences in legal systems, security breaches in the ASP's infrastructure, finally, a virtual inability of the data owner to detect and prevent any misuse of the outsourced data limit the applicability of a legislative regulation. This raises the following question: how can we ensure security, integrity and private access for the data that should be transferred to the ASP as a service input allowing the ASP to efficiently provide this service?

The thesis considers the possibilities of secure outsourcing of databases and of content-based routing operations to the ASP that is not trusted. We explore the limits of the security that is achievable in these scenarios while preserving their practical applicability and efficiency. When talking about security and efficiency we refer to the state of the art definitions from cryptography and complexity theory. This allows us to use a formal and rigorous approach for examining conditions under which certain levels of security are achievable as well as what cannot be achieved without violating security and efficiency requirements.

Also, apart from discussing *what* can be done to ensure the security of the service input data, this thesis provides practical results that show *how* security of the outsourced data can be ensured. We propose techniques that enable privacy-preserving database outsourcing by allowing us to run certain SQL queries against securely encrypted database without or with minimal information leaks to the ASP, to ensure integrity of the outsourced data and to define rules for a discretionary access to it. We also consider a problem of secure content-based routing and propose an approach that allows for performing such routing without revealing any information about the routed messages to the party performing the routing, represented by a distributed network of routing nodes.

1.2 Contributions

These are the main contributions of the thesis:

- *Secure outsourcing of databases to untrusted service providers using privacy homomorphisms and searchable encryption techniques*

  We explore the applicability of special cryptographic constructs that allow performing operations over encrypted data, also known as *privacy homomorphisms*, for creating protocols that could allow secure database outsourcing to a non-trusted service provider. We also describe a general framework for secure database outsourcing that is based on encryption schemes that allow for performing search on encrypted data and formally prove its correctness and security.

- *A new searchable encryption scheme*

  We describe a new encryption schemes that allows for performing search on encrypted data and has characteristics superior to those of existing counterparts. We formally prove its correctness and security. Furthermore, the proposed scheme allows for additional functionality when applied to the framework for secure database outsourcing.

- *Managing discretionary access to securely outsourced and encrypted databases*

  We propose an approach for managing discretionary access to securely outsourced and encrypted databases. Compared to the existing techniques, most of which are applicable only when users form hierarchy according to their permissions, our approach is applicable for general scenarios, is simpler and has similar performance characteristics.

- *Limitations and possibilities for secure content-based routing in non-trusted environment*

  We examine possibilities of a secure content-based routing performed by non-trusted routing nodes. We build a formal security model that describes a secure content-based routing system, evaluate existing approaches against this model and provide a formal analysis of the possibilities for achieving confidentiality of routed content for various network structures.

1.3 Structure of the Thesis

The rest of the thesis is structured as follows:

Chapter 2 describes the notation and introduces the basic concepts of cryptography. We also provide an introduction to the notion of provable security, which is used frequently throughout the thesis.

Chapter 3 discusses the problem of secure database outsourcing. We elaborate on a formal security model and describe a framework that, using a similarity between full-text search and some database operations, allows for constructing schemes that enable secure database outsourcing within the proposed model. Furthermore, we present a new searchable encryption scheme that exceeds existing analogues with regard to certain parameters and extends the applicability of the proposed secure database outsourcing framework.

Chapter 4 considers the problem of providing a discretionary access control to a securely outsourced database. We describe a solution that enables read/write access rules without any involvement of the database service provider. We examine practicability of the solution by evaluating results of conducted performance experiments against a possible application scenario.

Chapter 5 concerns with the possibility of a secure content-based routing in a non-trusted environment. We introduce several security models that provide a formal treatment for the notion of confidential routing. Furthermore,

we explore possibilities of providing solutions that could enable confidentiality according to these models and allows performing the routing efficiently.

Results presented in Chapter 2 were published in [EFG06; EG07a] and results presented in Chapter 4 were published in [EG07b].

Chapter 2

Preliminaries

In this chapter, we familiarize the reader with the terminology used throughout the thesis. We introduce the relevant notation and elaborate on basic cryptography constructs and definitions. Additionally, we provide a brief introduction to the provable security paradigm, which serves as a basis for discussing security characteristics of our solutions.

2.1 Notation

By $\{0,1\}^n$ we define the set of all binary strings of length n. Sometimes the notation 1^n is used for defining a n-bit string in unary notation $(\underbrace{1,\ldots,1}_{n})$. The bit length of string a is denoted as $|a|$. We use the notation $a|b$ to denote a concatenation of strings a and b. To show that element k is randomly and uniformly chosen from set $\mathcal{K}$ we write $k \xleftarrow{R} \mathcal{K}$.

By $\Pr[A]$ we define a *probability* of event A. *Conditional probability* $\Pr[A|B]$ is the probability of event A assuming that the event B has taken place. Given the probability of events A and B both taking place ($\Pr[A \cap B]$) the conditional probability can be expressed as

$$\Pr[A|B] = \frac{\Pr[A \cap B]}{\Pr[B]}$$

The law of total probability allows to rewrite the probability of event A as

$$\Pr[A] = \sum_i \Pr[A \cap B_i] = \sum_i \Pr[A|B_i]\Pr[B_i]$$

where $\{B_i | i = 1, \ldots\}$ is a partition of a probability space (finite or countably infinite).

By $\Pr[A(x_1, \ldots, x_t) = \alpha]$ we define a probability that algorithm A given strings $x_1, \ldots, x_t$ as its input outputs α. The *advantage* of (adversarial) algorithm A (defined as $\mathrm{Adv}A$) shows how good the algorithm is in guessing which of two string x or y it received as an input:

$$\mathrm{Adv}A = |\Pr[A(x) = 1] - \Pr[A(y) = 1]|$$

A function $f : \mathbb{R} \mapsto \mathbb{R}$ is said to be *negligible* if it decreases faster than one over any polynomial, meaning that for every positive polynomial $p(\cdot)$ and all sufficiently large n,

$$f(n) < \frac{1}{p(n)}$$

where the phrase "for all sufficiently large n" means "$\exists N \, \forall n > N$".

A function $f : \mathbb{R} \mapsto \mathbb{R}$ is said to be *significant* if there is a positive polynomial $p(\cdot)$ such that there is always a sufficiently large n that

$$f(n) \geq \frac{1}{p(n)}$$

where the phrase "there is always a sufficiently large n" means "$\forall N \, \exists n > N$".

2.2 Computation Model

An *algorithm* is a Turing machine that when given an input halts after a finite number of steps. The input is provided via an *input* tape and the result is written on an *output* tape. An algorithm is *probabilistic* if it has an additional input tape filled with randomly and uniformly distributed zeroes and ones. Alternatively, a probabilistic algorithm can be modeled as a Turing machine that is able to generate additional random input by throwing a *fair coin*.

It is said that an algorithm is *polynomial time* if when given any string $x \in \{0,1\}^n$ it halts after $p(n)$ steps where $p(\cdot)$ is a positive polynomial. It is generally accepted that polynomial time algorithms are *feasible* or fast enough to be considered practical, as opposed to super-polynomial time algorithms that are considered impractical requiring, for example, an exponential number of steps.

An algorithm may have access to one or more *oracles*. The notion of the oracle allows modeling an ability of the algorithm to ask questions to the outside, receiving well-defined responses. The oracle provides the algorithm with the possibility for performing computations that it cannot do on its

own. Formally, the oracle is defined as additional input and output tapes. The algorithm uses the oracle input tape for writing an *oracle query* (input for the "outside" computation) and the result of the query is written to the oracle output tape. The reply to each oracle query is given in a single step. The algorithm with access to oracles $O_1, \ldots, O_l$ is defined as $A^{O_1,\ldots,O_l}$.

2.3 Random Functions and Permutations

Consider a family of functions $\Phi : \mathcal{K} \times \mathcal{X} \mapsto \mathcal{Y}$ where $\mathcal{K} = \{0,1\}^n$ is a set of *keys*, $\mathcal{X} = \{0,1\}^m$ is the domain of Φ and $\mathcal{Y} = \{0,1\}^l$ is the range of Φ. Key $k \in \mathcal{K}$ defines a function $\Phi_k(x) = \Phi(k,x)$, which is an instance of family Φ. Keys that are assigned to each instance allow to interpret family Φ as a random variable. If a random variable Φ has a uniform distribution then we say that Φ is a *random function*, meaning that an instance function Φ_k is randomly and uniformly drawn from a family of all functions, mapping all binary strings of length l to binary strings of length m.

Analogously, one can define a *random permutation*. If $\Psi : \mathcal{K} \times \mathcal{X} \mapsto \mathcal{X}$ where $\mathcal{K} = \{0,1\}^n$ and $\mathcal{X} = \{0,1\}^m$ is a family of permutations $\Psi_k(x) = \Psi(k,x)$ and random variable Ψ is uniformly distributed, then we say that Ψ is a *random permutation*.

2.4 Cryptographic Primitives

A cryptographic primitive is a basic cryptographic algorithm that is used as a building block for constructing more complex security schemes possessing certain functional properties and satisfying certain security requirements. We introduce cryptographic primitives essential for this thesis referring to standard cryptography definitions; see, example, [Gol01],[Gol04].

A *pseudo-random function* is a family of functions, such that a randomly chosen instance of that family is indistinguishable by its output/input behavior from an instance drawn from a random function:

Definition 2.4.1 (pseudo-random function)**.** *A mapping $F : \mathcal{K} \times \mathcal{X} \mapsto \mathcal{Y}$, where $\mathcal{K} = \{0,1\}^n$, $\mathcal{X} = \{0,1\}^m$, $\mathcal{Y} = \{0,1\}^l$ is a pseudo-random function if for every PPT oracle algorithm A, every positive polynomial $p(\cdot)$, and all sufficiently large n,*

$$|\Pr[A^{F_k} = 1] - \Pr[A^{\Phi_k} = 1]| < \frac{1}{p(n)}$$

where Φ_k is an instance of random function $\Phi : \mathcal{K} \times \mathcal{X} \mapsto \mathcal{Y}$.

A pseudo-random function can be also described as a family of functions for which the advantage to be distinguished from a random function is negligible.

Similarly, one can define a *pseudo-random permutation*:

Definition 2.4.2 (pseudo-random permutation)**.** *A mapping $P : \mathcal{K} \times \mathcal{X} \mapsto \mathcal{X}$, where $\mathcal{K} = \{0,1\}^n$, $\mathcal{X} = \{0,1\}^l$ and $P(k,\cdot)$ is a permutation for all $k \in \mathcal{K}$ is a pseudo-random permutation if for every PPT oracle algorithm A, every positive polynomial $p(\cdot)$, and all sufficiently large n,*

$$|\Pr[A^{P_k} = 1] - \Pr[A^{\Psi_k} = 1]| < \frac{1}{p(n)}$$

where Ψ_k is an instance of random permutation $\Psi : \mathcal{K} \times \mathcal{X} \mapsto \mathcal{X}$.

Consider now an environment consisting of a *sender*, a *receiver* and an *adversary*. If the sender wants to send a message to the receiver but prevent the adversary from learning the content of this message, it should apply a secret transformation that maps the original *plaintext* message to the corresponding *ciphertext* and the transformation should be efficiently invertible by the receiver. Such a transformation is called an *encryption algorithm*, while the inverse transformation is called a *decryption algorithm*. Additionally, the sender and the receiver should share a secret parameter also known as a *secret key* that is provided as an additional input to the encryption and decryption algorithms. Complying with *Kerckhoffs' principle* which states that in a secure system the only secret parameter should be a key [Ker83] and which is generally accepted by cryptographers, we assume that the encryption and the decryption algorithms are known to all the parties, while the key is secretly shared between the sender and the receiver. Altogether the set of possible keys, the encryption and decryption algorithms constitute an *encryption scheme*. An additional requirement for an encryption scheme that helps to ensure that the adversary cannot easily guess which key was picked up by the sender and the receiver is to have the key randomly and uniformly chosen from the set of its possible values.

To describe it formally, we define a set of plaintexts as $\mathcal{X} = \{0,1\}^m$, a set of ciphertexts as $\mathcal{C} = \{0,1\}^l$ and set of keys $\mathcal{K} = \{0,1\}^n$.

Definition 2.4.3 (symmetric encryption scheme)**.** *A symmetric encryption scheme is a triple $(\mathcal{K}, E, D)$, where $E : \mathcal{K} \times \mathcal{X} \mapsto \mathcal{C}$ is a PPT algorithm (encryption algorithm) that maps a key $k \in \mathcal{K}$ and a plaintext $x \in \mathcal{X}$ into a corresponding ciphertext $c \in \mathcal{C}$ and $D : \mathcal{K} \times \mathcal{C} \mapsto \mathcal{C}$ is a PPT algorithm (decryption algorithm) that maps a key k and a ciphertext c into a corresponding plaintext x. It must hold that $D_k(E_k(x)) = x$. The key is chosen*

randomly and uniformly from the key space $\mathcal{K}$. The bit length n of the keys is called security parameter of the scheme.

It is clear that the size of the key space $\mathcal{K}$ that is defined by n plays a critical role in the security of the scheme. If n is too small, since encryption and decryption algorithms are public knowledge, the adversary can break the scheme (map a ciphertext to the corresponding plaintext) by simply trying all possible keys. Such a type of attack is also known as a *brute force attack*. Keeping n large enough is important for being able to withstand this type of attack, since the attacker will have to try 2^n possible keys in the worst case and 2^{n-1} on average. That is why n is often called a *security parameter* of the encryption scheme.

There are numerous implementations of the symmetric encryption schemes such as Data Encryption Standard (DES) [Nat77], Advanced Encryption Standard (AES) [Nat01], Twofish [SKW+98].

While symmetric encryption schemes require that parties willing to establish a secure communication agree on a secret key beforehand, there exists a class of encryption schemes that allow to avoid this extra step by having two separate keys for encryption and decryption algorithms: a key that is used for encryption and is a public information – *public key*, and a key that is used for decrypting messages encrypted by the corresponding public key and is kept secret – *private key*. In order to be able to receive encrypted messages, the receiver generates a public and a private keys making the public key freely accessible. A party (the sender) willing to securely send a message to the receiver encrypts it, using the encryption algorithm and the receiver's public key, and sends the produced ciphertext to the receiver. The receiver uses its private key and the decryption algorithm for decrypting the received ciphertext.

Encryption schemes that allow for performing such communication are called *asymmetric* (or *public-key*) *encryption schemes*:

Definition 2.4.4 (asymmetric encryption scheme)**.** *An asymmetric encryption scheme is a triple (G, E, D), where $G : \mathbb{N} \mapsto \mathcal{K}_e \times \mathcal{K}_d$ is a key-generating algorithm that given a security parameter n generates a pair of encryption and decryption keys $G(n) = (k_e, k_d)$, $E : \mathcal{K}_e \times \mathcal{X} \mapsto \mathcal{C}$ is a PPT algorithm (encryption algorithm) that maps a public key $k_e \in \mathcal{K}_e$ and a plaintext $x \in \mathcal{X}$ into a corresponding ciphertext $c \in \mathcal{C}$ and $D : \mathcal{K}_d \times \mathcal{C} \mapsto \mathcal{C}$ is a PPT algorithm (decryption algorithm) that maps a private key $k_d \in \mathcal{K}_d$ and a ciphertext c into a corresponding plaintext x. It must hold that $D_{k_d}(E_{k_e}(x)) = x$.*

Examples of asymmetric encryption schemes are RSA [RSA78], ElGamal [Gam84], Cramer-Shoup [CS98].

Compared to symmetric schemes where the key is randomly and uniformly chosen from set $\{0,1\}^n$, asymmetric schemes generate keys in a more elaborate way. That is captured by introducing a key-generating algorithm that takes a security parameter, which does not necessarily define the length of the keys, as an input and outputs a pair of public and private keys.

Furthermore, when talking about symmetric encryption schemes we will be simply referring to them as to encryption schemes, as opposed to asymmetric encryption schemes.

2.5 Provable Security

Describing a structure of an encryption scheme Definition 2.4.3 says nothing about its security. Consider the following example:

$$\begin{aligned} \mathcal{K} &= \{0,1\} \\ \forall k\, E_k(x) &= x \\ \forall k\, D_k(c) &= c \end{aligned}$$

Although this construction perfectly satisfies Definition 2.4.3, it is obvious that no matter how large n is, it provides no security at all.

In order to be able to provide a formal treatment for a notion of security of an encryption scheme, one should formally define goals of the participating parties (the sender, the receiver and the adversary), their computation capabilities and resources they can access. As mentioned above, a basic goal of the sender and the receiver is to be able to exchange messages, keeping their content secret from the adversary. The goal of the adversary, in contrast, is to be able to read the content of the transmitted messages. We assume that the receiver has a way of receiving all messages sent by the sender and that the adversary has the ability to eavesdrop on all the exchanged messages. We also assume that the sender, the receiver and the adversary have bounded computational capabilities. To model this we refer to the notion of a probabilistic polynomial-time (PPT) algorithm: the sender is modeled by the PPT encryption algorithm E, the receiver is modeled by the PPT decryption algorithm D and the adversary is modeled by the PPT adversarial algorithm P.

The most obvious, albeit not correct way to define security of encryption scheme $(\mathcal{K}, E, D)$ is to require that the adversary, given ciphertext $c = E_k(x)$, where k is randomly and uniformly chosen from $\mathcal{K}$, and without any knowledge about key k, is unable to correctly guess the original plaintext x. Since the adversary is modeled as a probabilistic algorithm, it can always try to

simply guess the key or the plaintext. Therefore, we substitute "could not guess" with "had a very small chance of guessing".

Substituting "very small" with "negligible," the aforementioned can be formalized by saying that for any computationally bounded adversary A the probability to guess x given only $E_k(x)$ is negligible: for every $x \in \mathcal{X}$, every PPT algorithm A, every positive polynomial $p(\cdot)$ and all sufficiently large n,

$$\Pr[A(E_k(x)) = x] < \frac{1}{p(n)}$$

However, when given a closer look, it becomes obvious that such an approach to defining security has major drawbacks. It does not say anything about the ability of the adversary to recover a half of the plaintext, neither does it say anything about whether the adversary, given several ciphertexts or a ciphertext and a plaintext, is able to perform less/greater comparisons or say whether the plaintext has even or odd number of zero bits, etc. Very often allowing adversary to answer such questions is enough to compromise the security of the whole system.

Therefore, we will rely on much stronger security definitions that allow us to describe an encryption scheme that can guarantee that a computationally bounded adversary given a ciphertext has extremely low chances of inferring even a single bit of information about the corresponding plaintext. The only information that the adversary might be allowed to recover is the length of the plaintext. Such a security model was first proposed in [GM84]. In the thesis, we will adhere to the notation introduced in [Gol04].

Definition 2.5.1 (semantic security)**.** *An encryption scheme* $(\mathcal{K}, E, D)$ *is* semantically secure *if for every PPT algorithm* A *there exists a PPT algorithm* A' *such that for any distribution of plaintexts represented by the random variable* χ*, every pair of polynomially bounded functions* $f(\cdot), h(\cdot)$*, randomly and uniformly chosen key* k*, every positive polynomial* $p(\cdot)$ *and all sufficiently large* n*,*

$$\Pr[A(E_k(\chi), 1^{|\chi|}, h(\chi)) = f(\chi)] < \Pr[A'(1^{|\chi|}, h(\chi)) = f(\chi)] + \frac{1}{p(n)}$$

The purpose of the function $h(\cdot)$ is to provide both algorithms with partial information about the plaintexts distribution χ. String $1^{|\chi|}$ provides the algorithms with the information about the length of the plaintext. Informally speaking, a knowledge of a ciphertext produced by a semantically secure encryption scheme does not significantly help the adversary in learning any property of the corresponding plaintext that is represented by the function $f(\cdot)$.

Perfectly serving its needs in defining a secure encryption scheme, the definition of semantic security is not very practical when it comes to showing whether or not an encryption scheme is secure or not. The following equivalent definition states that a secure encryption scheme should not allow a computationally-bound adversary to distinguish between encryptions of two plaintexts.

Definition 2.5.2 (indistinguishable security)**.** *An encryption scheme* $(\mathcal{K}, E, D)$ *is indistinguishably secure if for every pair of plaintexts* x_1, x_2*, every PPT algorithm* A*, randomly and uniformly chosen keys* k*, every positive polynomial* $p(\cdot)$*, and all sufficiently large* n*,*

$$|\Pr[A(E_k(x_1)) = 1] - \Pr[A(E_k(x_2)) = 1]| < \frac{1}{p(n)}$$

The proof of the equivalence of Definitions 2.5.1 and 2.5.2 can be found in [Gol04].

Definitions 2.5.1 and 2.5.2 guarantee security only if the sender and the receiver exchange a single message. However, that does not automatically mean that the sender and the receiver can use an encryption scheme that satisfies these definitions for securely exchanging several messages by encrypting them with the same key. A classical example of an encryption scheme that is secure in the sense of Definitions 2.5.1 and 2.5.2 but starts leaking information when used for encrypting multiple plaintexts is *one-time pad*. With one-time pad, the encryption is performed by XORing a plaintext with a key that should be of the same length as the plaintext: $c = x \oplus k$. Hide all information about a plaintext from a computationally unbound adversary [Sha49], when used for encrypting several messages with the same key XORing the corresponding ciphertexts reveals at which positions the plaintexts have the same bits and at which positions the bits are different. E.g., let $x_1 = (011)$, $x_2 = (110)$ and $k = (100)$. Then $c_1 = (011) \oplus (100) = (111)$, $c_2 = (110) \oplus (100) = (010)$ and $c_1 \oplus c_2 = x_1 \oplus k \oplus x_2 \oplus k = x_1 \oplus x_2 = (111) \oplus (010) = (010)$ where 1 at i-th position means that i-th bits x_1 and x_2 are different and 0 means that the corresponding bits of x_1 and x_2 are the same.

Another problem that makes Definitions 2.5.1 and 2.5.2 quite impractical is that they allow encryption schemes to be deterministic, thus encrypting the same plaintexts as the same ciphertexts (e.g., one-time pad): $E_k(x_1) = E_k(x_2)$ if $x_1 = x_2$. If a sufficient number of plaintexts is encrypted with such a scheme the adversary, by comparing the frequencies of the resulting ciphertexts with an a-priori known distribution of the plaintexts, can recover some of the plaintext values. Such a type of attack is also known as *statistical attack*.

Encryption schemes that allow using the same key for encrypting more than one message and guarantee that chances of leaking any information about the encrypted messages are negligibly small should satisfy stronger variants of Definitions 2.5.1 and 2.5.2 – semantic and indistinguishable security for multiple messages. Since in all our proofs we will rely only on definitions of indistinguishable security, we omit their semantic counterparts. Interested readers may consult [Gol04] for details.

Let $\bar{x} = (x^{(1)}, \ldots, x^{(q)})$ and $\bar{E}_k(\bar{x}) = (E_k(x^{(1)}), \ldots, E_k(x^{(q)}))$, $q = q(n) \leq poly(n)$, where $poly(\cdot)$ is an arbitrary positive polynomial.

Definition 2.5.3 (indistinguishable security for multiple messages)**.** *An encryption scheme $(\mathcal{K}, E, D)$ is* indistinguishably secure multiple messages *if for every pair of plaintext sequences $\bar{x}_1 = (x^{(1)}, \ldots, x^{(q)}), \bar{x}_2 = (y^{(1)}, \ldots, y^{(q)})$, $q = q(n) \leq poly(n)$, every PPT algorithm A, randomly and uniformly chosen keys k, every positive polynomial $p(\cdot)$, and all sufficiently large n,*

$$|\Pr[A(\bar{E}_k(\bar{x}_1)) = 1] - \Pr[A(\bar{E}_k(\bar{x}_2)) = 1]| < \frac{1}{p(n)}$$

The definitions of indistinguishability provided so far guarantee the protection only from a "passive" adversary. Such an adversary simply eavesdrops on ciphertexts and analyzes them, trying to obtain some information about the corresponding plaintexts. But in real applications, the adversary can also be "active" and additionally trick the sender into encrypting messages of her choice (chosen-plaintext attack) or even cause the receiver to decrypt ciphertexts of her choice (chosen-ciphertext attack). It is modeled as the ability of the adversarial algorithm to query encryption and decryption oracles. It may seem that the assumption of an adversary's ability to encrypt and decrypt ciphertexts of her choice is very unlikely to be satisfied. However, the successful chosen-ciphertext attack on the widely used Internet security protocol SSL discovered by Bleichenbacher [Ble98] demonstrates the relevancy of such scenarios.

While performing a chosen-ciphertext attack, the adversary has access only to the encryption oracle.

Definition 2.5.4 (indistinguishable security under chosen-plaintext attack)**.** *An encryption scheme $(\mathcal{K}, E, D)$ is indistinguishably secure under chosen plaintext attack if for every pair of plaintexts x_1, x_2, every PPT oracle algorithm A^{E_k}, randomly and uniformly chosen keys k, every positive polynomial $p(\cdot)$, and all sufficiently large n,*

$$|\Pr[A^{E_k}(E_k(x_1)) = 1] - \Pr[A^{E_k}(E_k(x_2)) = 1]| < \frac{1}{p(n)}$$

In [Gol04] it is shown that Definitions 2.5.3 and 2.5.4 are equivalent.

While performing a chosen-ciphertext attack, the adversary also has access to the decryption oracle. Usually, in scenarios where a chosen-ciphertext attack is possible, a chosen-plaintext attack is possible, too. To capture this notion, we allow the adversary to query the encryption oracle as well.

Definition 2.5.5 (indistinguishability under chosen-ciphertext attack)**.** *An encryption scheme $(\mathcal{K}, E, D)$ is indistinguishable under chosen-ciphertext attack if for every pair of plaintexts x_1, x_2, every PPT oracle algorithm A^{E_k, D_k}, randomly and uniformly chosen keys k, every positive polynomial $p(\cdot)$, and all sufficiently large n,*

$$|\Pr\left[A^{E_k,D_k}(E_k(x_1)) = 1\right] - \Pr\left[A^{E_k,D_k}(E_k(x_2)) = 1\right]| < \frac{1}{p(n)}$$

Definitions 2.5.3, 2.5.4, 2.5.5 imply that an encryption scheme satisfying either of these definitions is probabilistic, encrypting the same plaintexts as different ciphertexts. Otherwise it would always be possible to distinguish a set of ciphertexts that are encryptions of the identical plaintexts from a set that contains encryptions of different plaintexts. On the contrary, pseudo-random functions and pseudo-random permutations are deterministic since it is required that they map identical plaintexts to identical ciphertexts.

Sometimes it is more convenient to define security of an encryption scheme as the following game played between an adversary and a sender:

1. The sender randomly and uniformly chooses key k from key space: $k \xleftarrow{R} \mathcal{K}$.

2. [chosen-plaintext attack] The adversary asks the decryption oracle for the plaintexts corresponding to the ciphertexts of her choice.

3. [chosen-ciphertext attack] The adversary asks the encryption oracle for the ciphertexts corresponding to the plaintexts of her choice.

4. The adversary chooses two plaintext sequences of the same length $\bar{x}_1 = (x^{(1)}, \ldots, x^{(q)}), \bar{x}_2 = (y^{(1)}, \ldots, y^{(q)}), q = q(n) \leq poly(n)$ and gives them to the sender.

5. The sender randomly and uniformly chooses β from set $\{1, 2\}$ and encrypts $\bar{x}_\beta$ using encryption algorithm E and key k.

6. The sender returns the resulting ciphertexts to the adversary.

7. [chosen-ciphertext attack] The adversary may additionally ask the oracle for the decryption of some ciphertexts except for the decryption of the one received from the sender.

8. The adversary tries to guess β.

9. If the adversary guesses β correctly, the outcome of the experiment is 1 and 0 otherwise.

If the probability that the PPT adversary can correctly guess bit β negligibly differs from 1/2, the adversary cannot obtain enough information about the plaintexts by querying the available oracles and observing the corresponding ciphertexts. This means that the encryption scheme does a good job hiding a content of the messages, or that the plaintexts are encrypted indistinguishably.

The experiment can be formally described as follows:

Experiment $\mathbf{Exp}_A$
$k \xleftarrow{R} \mathcal{K} = \{0,1\}^n$
Let $\bar{x}_1 = (x^{(1)}, \ldots, x^{(q)})$ and $\bar{x}_2 = (y^{(1)}, \ldots, y^{(q)}), q = q(n) \leq poly(n)$
$\beta \xleftarrow{R} \{0,1\}$
$g \leftarrow A(\bar{E}_k(\bar{x}_\beta))$, where $\bar{E}_k(\bar{x}) = (E_k(x^{(1)}), \ldots, E_k(x^{(q)}))$
if $\beta = g$ **return** 1
else return 0

Definition 2.5.6 (Indistinguishable security)**.** *Encryption system* $(\mathcal{K}, E, D)$ *is indistinguishably secure if for every PPT A, every positive polynomial* $p(\cdot)$*, all sufficiently large n the probability of* $\mathbf{Exp}_A$ *returning 1 cannot be significantly greater than the probability of guessing* β *by drawing it randomly and uniformly from set* $\{1,2\}$*:*

$$\Pr[\mathbf{Exp}_A = 1] - \frac{1}{2} < \frac{1}{p(n)}$$

It is relatively easy to prove the equivalence of definitions 2.5.2 and 2.5.6.

Theorem 1. *Definitions 2.5.3 and 2.5.6 are equivalent.*

Proof.
I. First we prove Definition 2.5.3 $\Rightarrow$ Definition 2.5.6 direction.

Assume that Definition 2.5.3 holds true but Definition 2.5.6 does not. Then there exists plaintext sequences $\bar{x}_1, \bar{x}_2$, a PPT algorithm A and a positive polynomial $p(\cdot)$ such that there always exists a sufficiently large n for which

$$\Pr[\mathbf{Exp}_A = 1] - \frac{1}{2} \geq \frac{1}{p(n)}$$

Using the algorithm A from Exp_A as a subroutine we construct a PPT algorithm B that hands its input to A and outputs 1 if A outputs 1 and 2 if A outputs 0:

Algorithm $B(\alpha)$
$1 : g \leftarrow A(\alpha)$
$2 :$ **if** $g = 1$ return 1
$3 :$ return 2

Then, by using the law of total probability we can rewrite the latter inequality as

$$\begin{gathered}
\Pr[\mathbf{Exp}_A = 1] - \frac{1}{2} = \\
\Pr[\mathbf{Exp}_A = 1|\beta = 1]\Pr[\beta = 1] + \Pr[\mathbf{Exp}_A = 1|\beta = 2]\Pr[\beta = 2] - \frac{1}{2} = \\
\frac{1}{2}\Pr[\mathbf{Exp}_A = 1|\beta = 1] + \frac{1}{2}\Pr[\mathbf{Exp}_A = 1|\beta = 2] - \frac{1}{2} = \\
\frac{1}{2}\Pr[B(\bar{E}_k(\bar{x}_1)) = 1] + \frac{1}{2}\Pr[B(\bar{E}_k(\bar{x}_2)) \neq 1] - \frac{1}{2} = \\
\frac{1}{2}\Pr[B(\bar{E}_k(\bar{x}_1)) = 1] + \frac{1}{2} - \frac{1}{2}\Pr[B(\bar{E}_k(\bar{x}_2)) = 1] - \frac{1}{2} = \\
\frac{1}{2}\Pr[B(\bar{E}_k(\bar{x}_1)) = 1] - \frac{1}{2}\Pr[B(\bar{E}_k(\bar{x}_2)) = 1] \geq \frac{1}{p(n)}
\end{gathered}$$

That contradicts to the assumption that Definition 2.5.3 holds true.

II. Analogously it can be shown that Definition 2.5.6 $\Rightarrow$ Definition 2.5.3.

Assume that Definition 2.5.6 holds true but Definition 2.5.2 does not. Then there exists plaintext sequences $\bar{x}_1, \bar{x}_2$, a PPT algorithm A and a positive polynomial $p(\cdot)$ such that there always exists a sufficiently large n for which

$$|\Pr[A(\bar{E}_k(\bar{x}_1)) = 1] - \Pr[A(\bar{E}_k(\bar{x}_2)) = 1]| \geq \frac{1}{p(n)}$$

Without losing the generality we assume that

$$\Pr[A(\bar{E}_k(\bar{x}_1)) = 1] - \Pr[A(\bar{E}_k(\bar{x}_2)) = 1] \geq \frac{1}{p(n)}$$

Then using the algorithm A as a subroutine we again construct a PPT algorithm B that hands its input to A and outputs 1 if A outputs 1 and 2 if the output of A is different from 1.

By using transformations analogous to those done in the first part of the proof the latter inequality can be rewritten as

$$\Pr[A(\bar{E}_k(\bar{x}_1)) = 1] - \Pr[A(\bar{E}_k(\bar{x}_2)) = 1] =$$
$$2 \cdot \Pr[\mathbf{Exp}_B = 1] - 1 \geq \frac{1}{p(n)}$$

what contradicts to the assumption that Definition 2.5.6 holds true.

□

In the thesis, we will refer only to the game-like variant of the indistinguishable security definition that considers a passive adversary. However, it is also straightforward to formulate similar definitions for "active" attacks and prove their equivalence to Definitions 2.5.4 and 2.5.5.

Furthermore, we do not provide definitions of security for asymmetric encryptions schemes since in the thesis we will only use the definitions relevant to symmetric cases. However, for asymmetric schemes the semantic and indistinguishable security can be defined analogously with the sole consideration that different keys are used for encryption and decryption, and the encryption key is known to the adversary.

2.6 Homomorphic Encryption

There is a class of encryption schemes that allow performing certain operations on plaintexts by performing corresponding (not necessarily the same) operations on ciphertexts. Such behavior makes encryption schemes possessing this property very similar to a *homomorphism* - a mapping between algebraic structures that preserves operations on these structures. As an example of a homomorphisms consider mapping $F : \mathbb{R} \mapsto \mathbb{R}$, $F(x) = 2x$. Observing that $F(a) + F(b) = 2a + 2b = 2(a + b) = F(a + b)$, we conclude that F is a homomorphism that preserves addition.

Since encryption is also a mapping from the set of plaintext to the set of ciphertexts, if an encryption scheme preserves some operations, it is natural to call it *privacy homomorphisms*.

Definition 2.6.1 (Privacy homomorphism)**.** *An encryption scheme* $(\mathcal{K}, E, D)$ *is a privacy homomorphism (*PH *for short), mapping a system of plaintexts* $(\mathcal{X}, \{\varphi_i\})$*, where* $\{\varphi_i\}$ *is a set of operations* $\varphi_i : \underbrace{\mathcal{X} \times \ldots \times \mathcal{X}}_{l_i} \mapsto \mathcal{X}$*, to a system of ciphertexts* $(\mathcal{C}, \{\psi_i\})$*, where* $\{\psi_i\}$ *is a set of operations* $\psi_i : \underbrace{\mathcal{C} \times \ldots \times \mathcal{C}}_{l_i} \mapsto \mathcal{C}$*, if for any* $x \in \mathcal{X}$*, the equation* $E_k(\varphi_i(x_1, \ldots, x_{l_i})) = \psi_i(E_k(x_1), \ldots, E_k(x_{l_i}))$ *holds.*

Privacy homomorphisms have been proposed for a number of operations, such as modulo arithmetic [RAD78], [DFHJ98] or full-text search on document sets [SWP00]. In the thesis we are mainly concerned with operations that could be applied to relational databases. In that case, $\mathcal{X}$ is a set of plaintext tables and $\{\varphi_i\}$ is a set of relational operations defined on these tables. Note that the set of operations can be confidential in their own right, i.e., it may be necessary to hide which operations should be performed and therefore, the transformation of $\{\varphi_i\}$ into $\{\psi_i\}$ might be also secret: i.e., an encryption operation. We call this transformation E^*.

Let $R = (a_1 : D_1, a_2 : D_2, \ldots, a_l : D_l)$ be a database schema, where a_i denotes an attribute name and D_i the domain of attribute a_i (all domains are finite). Then,

$$\mathrm{Tup}(R) = \{\langle a_1 : d_{k1}, \ldots, a_l : d_{kl}\rangle | d_{ki} \in D_i,\ a_i \neq a_j,\ i \neq j\},$$

is the set of all possible data tuples allowed by R, where k is the tuple index and i, j are column numbers. Every subset $T(R) \subseteq \mathrm{Tup}(R)$ is called a *table* fitting schema R. When it is clear from a context which schema is implied, we simply write T. $\mathbb{R} = (\mathrm{Tup}(R))^s$ is the set of all possible tables of R, i.e., the power set of $\mathrm{Tup}(R)$. Let $\{\varphi_i\}$ be a set of relational operations. Then a database PH can be defined as follows:

Definition 2.6.2 (Database privacy homomorphism)**.** *A database privacy homomorphism is a PH* $(\mathcal{K}, E, E^*, D)$ *where* $E : K \times \mathbb{R} \mapsto \mathbb{C}$ *encrypts tables,* $E^* : K \times \{\varphi_i\} \mapsto \{\psi_i\}$ *encrypts queries, and* $D : K \times \mathbb{C} \mapsto \mathbb{R}$ *decrypts tables.*

The transformations E and E^* draw a single key from the set $\mathcal{K}$. If the two algorithms require independent keys, we simply define them to use different parts of the bit representation of a given key. If they require keys that are functionally depend upon each other, then we can encode this function dependence in the algorithms as well.

Chapter 3

Secure Database Outsourcing

In this chapter, we consider the problem of secure database outsourcing. We start with conducting an analysis of functional and security requirements of a system providing secure database outsourcing. We then describe a solution that allows for secure database outsourcing and is based upon encryption schemes that enable a search on encrypted data. Finally, we propose a new encryption scheme that enables a search on encrypted data, surpasses existing analogues with regard to certain characteristics and allows for extended functionality when used for secure database outsourcing.

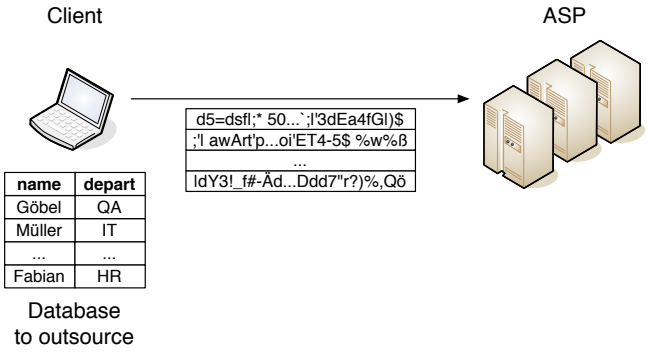
3.1 Introduction

In this section, we describe the problem of secure database outsourcing, review related work and briefly list our main results related to this problem.

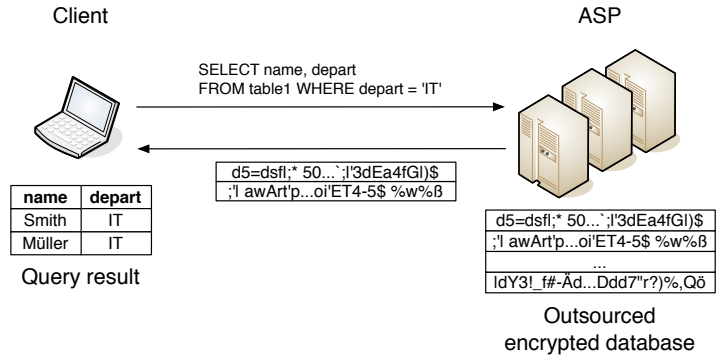
3.1.1 Motivation and Problem Statement

Consider a 2-party outsourcing problem in which a client wants to outsource database operations on sensitive data sets to an ASP without having to trust it. Forming contracts and relying on law enforcement are options, but for reasons that we mention below their effectiveness is limited. However, the costs of negotiations, auditing and prosecution can be considerable [BG02]. To protect the outsourced data, the client would rather encrypt it in a way that enables the ASP to perform operations on the ciphertext yielding encrypted results, which the client could in turn decrypt. All this should ideally take place without revealing anything about the plaintext data or the performed operations to the ASP (Figures 3.1a - 3.1b).

Consider the following examples:



(a) Outsourcing database



(b) Querying database

Figure 3.1: Secure database outsourcing

1. In case the ASP changes an owner, it might become unclear whether the new owner is still legally bound by the initial contract where privacy policies are defined [Dis00; San00]. As Amazon.com states in its privacy notice: "*Also, in the unlikely event that Amazon.com, Inc., or substantially all of its assets are acquired, customer information will of course be one of the transferred assets*" [Ama08]. However, if the data is not given away as a cleartext, such an issue will not even arise.

2. Storing and processing emails on a remote server is in many respects similar to operating a remote database. A user that stores emails on a remote server is not protected from having her email read by someone else [Zet04]. Being able to store emails in an encrypted form in such a way that the email service still remains usable would eliminate this risk completely.

3. German Federal Data Protection Act [Bds02] explicitly imposes severe

limitations on any transfer of personal data to third countries. According to the Act, such data transfer is permissible only if "an adequate level of protection" is provided. If there were an appropriately certified solution providing means for secure database outsourcing, it might have been considered "adequate level of protection".

The provided examples clearly illustrate the usefulness of secure outsourcing approach. However, the ability to perform meaningful processing of the securely encrypted data seems counterintuitive at first glance. Indeed, the sole purpose of encryption is to hide content of the data from anyone who does not possess the key. Clearly, if the ASP is not trusted, it should not be provided with the key, thus raising the questions as to how could it possibly perform any meaningful data processing without being able to interpret it. However, as it will be shown, by using specially constructed encryption schemes, some practically meaningful processing is still achievable. Moreover, the characteristics of such schemes can be formally evaluated allowing us to make theoretically sound statements about their security.

3.1.2 Related Work

The idea that it is possible to use privacy homomorphisms for secure database outsourcing was first expressed in 1978 by Rivest et al. [RAD78]. If the privacy homomorphism preserved some of the relational operations, then it would be possible to process encrypted relations without decrypting them. As a trivial example, consider an encryption scheme that tuple by tuple deterministically encrypts all the attribute values of the database tables. Deterministic encryption means that each plaintext is bijectively mapped to the corresponding ciphertext. That allows us to state that equality of the ciphertexts means equality of the corresponding plaintexts and, therefore, if the whole database is encrypted with such an encryption scheme, it is possible to perform exact selects, unions, differences, Cartesian products and projections on the encrypted tables. Unfortunately, such a straightforward solution is vulnerable to statistical attacks and cannot be considered for any practical use.

In 2001 Hacıgümüş et al. [HILM02] described an encryption scheme that allowed for performing all relational operations on an encrypted database and rendered the statistical attack on the scheme less obvious than in the example described above. According to the scheme, the domain of each attribute is partitioned into intervals, and each attribute value is mapped to the interval that contains it. The intervals are then deterministically encrypted and attached to the secure encryptions of the tuples. The way the relational

operations are carried out is similar to the deterministic database privacy homomorphism we have just described. The only difference is that instead of operating with deterministically encrypted attribute values, the scheme uses the deterministically encrypted containing intervals, while keeping the attributes securely hidden. Thus, for example, an exact select operation will return the tuples with the attribute values contained in the interval that is stated as the argument of the select operation. On the one hand, this requires the client to perform postfiltering in order to remove the tuples that have the attribute values belonging to the queried interval and are not equal to the argument of the select operation. On the other hand, it renders the attack on the encryption scheme less straightforward. However, it is clear that the ASP nevertheless learns *something* about the data.

A modification of this scheme was proposed in [DVJ+03] where, additionally, exposure coefficients were computed, measuring how much information is given away, thus allowing balancing the efficiency with the confidentiality. However, there is still no clear notion of what amount of information leakage is too much for the database owner, and whether the scheme or its variants can fulfill their requirements.

In [BG03] the authors propose to directly apply a privacy homomorphism that preserves certain arithmetic operations [DFHJ98], allowing for performing these operations over encrypted data stored in the outsourced database. This allows for processing SELECT queries containing corresponding arithmetic operations on the server-side without any post-processing on the client-side. However, in order to be able to process equality condition of SELECT queries, the original privacy homomorphism is modified to deterministically encrypt plaintexts. As we already mentioned, it dramatically affects security of the whole system by making it vulnerable to the statistical attack.

A promising approach that might help in designing practical database privacy homomorphisms is to consider algorithms for secure full-text keyword search on encrypted data (referred to as *searchable encryption schemes* in the following) [BCOP04; CM05; Goh03; SWP00; YZW06]. These schemes, being indistinguishably secure, allow for performing searches on encrypted data. One can refer the necessity to perform a postfiltering of the search results to limitations of the schemes proposed in [Goh03; SWP00], since these schemes allow erroneous tuples to be included in the result of a search operation with a high probability. The schemes proposed in [BCOP04; CM05] are able to perform search only for a fixed set of keywords, which also limits their applicability. An encryption scheme similar to the one we propose in the thesis is described in [YZW06]. However, the analytical part of this research contains several serious flaws. Thus, as it can be illustrated by a counterexample, the definition of security on which the authors base their

reasoning does not require a database to be encrypted at all. Additionally, the authors mistakenly suppose that their scheme does not include erroneous tuples in the resulting set of a processed query. In Section 3.4.6, we provide a detailed analysis of this research.

Another interesting direction of research is Private Information Retrieval (PIR). The idea of PIR is to allow the client to query the outsourced database revealing neither queries, nor their results. Compared to the problem of secure database outsourcing, the content of the database is not required to be kept secret from the ASP. In [CGKS95] it was shown that PIR is achievable only if the database is replicated between multiple, non colluding servers. A more realistic and practical variation of PIR problem is *computationally Private Information Retrieval* (cPIR) that considers a computationally bounded adversary. A number of protocols implementing cPIR without the replication requirement and any support from trusted parties have been proposed in the literature [CG97; KO97; CMS99]. Unfortunately, as was pointed in [SC07], the computational and communicative overheads of the existing protocols are still too significant for considering them as practical since having the client download the entire database and run queries on the downloaded copy appears to be more efficient. Another approach to implement PIR efficiently is to rely on a trusted physical device (*secure coprocessor*), which is available off-shelf [ABCS06] and provides a trusted environment for performing secure computations. Under that assumption, [AF02; IS05; WDDB06] propose practically applicable PIR protocols with $O(1)$ computational and optimal communicative complexities at the cost of periodical preprocessing of the database.

3.1.3 Our Contribution

In this chapter, we define a security model for a 2-party database outsourcing scenario that transforms relational data sets and queries into ciphertexts such that (*i*) the data is *securely hidden* from the ASP, although the ASP has unlimited access to the ciphertext; and (*ii*) given encrypted queries the ASP can compute ciphertext results which the client (the database owner) can *efficiently* decrypt. We will capture these properties in formal definitions and perform their rigorous analysis, obtaining the conditions under which they can be satisfied.

As we explore these issues, we find that database privacy homomorphisms that could satisfy these conditions are rare indeed, and that many reasonable and highly desirable security notions are extremely hard to meet by any possible encryption scheme. For example, it seems to be very problematic to design an efficient protocol that could prevent the ASP from making in-

ferences about encrypted data by analyzing results of queries issued by the client. Fortunately, there are more relaxed scenarios that still have important practical applications. In particular, the client may be comfortable with merely ensuring that if the ASP turns malicious *after* (not *while*) processing his data, it has no way of decrypting the ciphertext. If it does become adversarial, the client can simply stop sending queries. The ASP then has no encrypted queries or encrypted query results to use in her assumed attack, and finding encryption algorithms for keeping the data safe becomes feasible. This security notion fits our "change of the owner" example nicely – change of ownership is a public process that can be easily witnessed by the client. We will present a class of database privacy homomorphisms suitable for this scenario, which are based on searchable encryption schemes and rigorously prove its security. Encryption schemes constituting this class preserve exact select relational operation and additionally allow insert, exact delete and union with duplicates operations.[1]

We also describe a novel searchable encryption scheme that can be directly applied as a database privacy homomorphism. In addition to the operations that can be supported when relying on the exiting searchable encryption schemes, our scheme also natively supports projection, Cartesian product and exact update.

The proposed encryption scheme displays the following key characteristics:

- The scheme is *provably secure* and can sustain a chosen-plaintext and a chosen-ciphertext attacks.
- The scheme reveals *nothing but the number of tuples that share a queried value* while performing an exact select.
- The scheme allows performing the supported operations on the encrypted database *efficiently*. It does not affect the time needed to perform projection, Cartesian product and insert operations. Checking whether a tuple satisfies an equality condition of an exact select requires $O(1)$ operations; therefore exact update, exact delete and exact select require $O(n)$ operations, where n is the number of tuples in the queried relation. The scheme also avoids a problem of many previous solutions, such as the outsourcing approach of [HILM02] or searchable encryption schemes that do not rely on a dictionary of allowed keywords

[1]Exact select, exact delete and exact update are variants of select, delete and update operations with condition predicates (WHERE-part of the corresponding SQL queries) restricted to a combination of equalities connected by AND or OR. The result of a union with duplicates is the union of two relations without duplicate tuples being removed.

[Goh03; SWP00]. All these solutions may return erroneous tuples that do not satisfy the query criteria. This requires the client to perform postfiltering of the received result set, which reduces the performance and complicates the development process of a client software. The only schemes that allow to perform search on encrypted data without requiring postfiltering are described in [BCOP04; CM05]. However, they can hardly be applied to databases since they support search for a set of predefined keywords, which constitutes a severe limitation. The scheme we are proposing may also include erroneous tuples in the result set of a search operation, but the probability of such an error is negligible.

3.2 A Security Analysis of Database Privacy Homomorphisms

In this section, we perform a security analysis of existing solutions for secure database outsourcing and identify main challenges that should be considered when building a system that enables secure outsourcing of databases.

3.2.1 Known Database Privacy Homomorphisms

For most existing database privacy homomorphisms, it is relatively easy to construct adversaries that win the game described in Definition 2.5.6. Consider the scheme proposed in [HILM02] and the following game. The adversary (the ASP) produces two (table, query) pairs. The tables are different, but the query is the same in both cases (for 4900 and 5400 choose any two values that do not belong to the same interval):

table 1:

ID	salary
171	4900
481	5400

table 2:

ID	salary
171	4900
481	4900

```
SELECT * FROM t WHERE salary = 4900
```

Both tables and queries are presented to the sender (the client) for encryption. The adversary obtains a ciphertext from the sender. Since the encryption is homomorphic, she can run the query on the table and examine the result. As each tuple is encrypted separately and, with high probability, only matching tuples are included in the result, the adversary can look at the size of the encrypted result: if the encrypted result is half the size of the encrypted table, then the adversary outputs 1, and 2 otherwise, making a

correct guess with probability 1. A similar attack works on the scheme of [DVJ+03].

To take another example, consider the scheme proposed in [BG03]. Again, the adversary maintains a table of encrypted salaries for the sender but this time the arithmetic mean of these salaries is of interest, not the unaggregated data itself. To this end, a privacy homomorphism for addition is used and the adversary computes the encrypted sum of all ciphertexts plus the number of ciphertexts, leaving the division to the sender. In order to launch an attack, the adversary can produce the same two tables as above, but this time the plaintexts are obtained by computing *two* queries on each table:

```
SELECT SUM(salary) FROM t
SELECT SUM(salary) FROM t WHERE salary = 4900
```

If the ciphertext results obtained by running these two queries are identical, then the adversary outputs 2; otherwise, she outputs 1. She wins with probability 1.

In fact, a much simpler attack can be successfully launched on this scheme. As all values of `salary` attribute are deterministically encrypted, the adversary can decide on her output simply by examining ciphertexts corresponding to the salaries: if they are identical, she outputs 2 and 1 otherwise.

3.2.2 Security Levels and Limitations

Database privacy homomorphisms are a subset of the set of all encryption schemes; therefore insights into the latter are also applicable to the former. However, the ability to transform plaintext operations into corresponding ciphertext operations yields new possibilities for an adversary to obtain insights into encrypted data. In this section, we will show that traditional definitions of security do not guarantee the secrecy of the data encrypted by a database privacy homomorphism, provide new definitions and explore how much security can and can not be achieved.

Consider a database privacy homomorphism $(\mathcal{K}, E, E^*, D)$ such that the encryption scheme $(\mathcal{K}, E, D)$ is indistinguishably secure but given a query φ the adversary is able to recognize it given $E_k^*(\varphi)$. Assume the adversary runs a query $E^*(\sigma_{a_i:d_i})$ on the encrypted table, where $\sigma_{a_i:d_i}$ is an exact select returning all tuples where a value of attribute a_i is equal to d_i. As result of the computation, the adversary obtains a set of encrypted tuples. Although she cannot decrypt them, she can infer that the value of the attribute a_i of these tuples is d_i merely from the fact that they are in the query result. Thus, while we used a rigorous security definition, our scheme is far less secure than the two we attacked in the previous section!

How is it possible that a database privacy homomorphism based on a

perfectly secure table encryption scheme can still leak so much information? What tricked us here is an ill-applied adversary model. The classical model does not consider that the adversary might be provided additional information when receiving and processing the queries. Our example has shown that these computations can cause unsuspected leaks. In order to capture this new situation, we need to craft new security definitions that are aware of the adversary's new capabilities. In other words, we need to take E^* into account. The example we discussed above makes it quite obvious that E^* should result in securely encrypted queries. However, the requirements for a database privacy homomorphism to provide indistinguishable encryptions for tables and queries along are not sufficient for preventing the adversary from learning private information about the database. This observation is illustrated in the following example.

Consider a database privacy homomorphism $(\mathcal{K}\times\mathcal{K}, E, E^*, D)$ that allows exact select queries. Let $(\mathcal{K}, E, D)$ be an indistinguishably secure encryption scheme, $E^* = E$ and (k_1, k_2) be a pair of keys randomly and uniformly chosen from the key space $\mathcal{K} \times \mathcal{K}$. Let $T = \{\langle x_1\rangle, \ldots, \langle x_m\rangle\}$ be a table consisting of one attribute and filled with m tuples. To encrypt table T, each of its attribute values is encrypted with key k_1 and the resulting set of ciphertexts is appended key k_2: $E_k(T) = (\{\langle E_{k_1}(x_1)\rangle, \ldots, \langle E_{k_1}(x_m)\rangle\}, k_2)$. To run an SQL query φ on encrypted table $E_k(T)$, the query is first padded up to a predefined for all queries length resulting in φ', encrypted as $\psi = E_{k_2}(\varphi')$, appended key k_1, and pair (ψ, k_1) is sent to the server. Upon receiving the query, the sever decrypts it using key k_2, which it was given as a part of $E_k(T)$, using key k_1, which was obtained with the query, the server decrypts $E_k(T)$, removes the padding from the resulting plaintext query and processes it on the plaintext database.

The indistinguishable security of encryption scheme $(\mathcal{K}, E, D)$ guarantees that any two encrypted tables of the same size as well as any two padded encrypted queries are indistinguishable. Thus, the proposed database privacy homomorphism satisfies the requirement for indistinguishable security of the table and queries encryptions. However, it is also obvious that this database privacy homomorphism provides no security at all since as soon as the first query is processed, the adversary obtains complete access to the data in plaintext.

Based on this insight, we propose the following improved definition (for simplicity we omit steps corresponding to chosen-plaintext and chosen-ciphertext attacks).

Consider the following experiment:

1. The sender randomly and uniformly chooses key k from key space:

$k \xleftarrow{R} \mathcal{K}$.

2. The adversary chooses two tables T_1, T_2 with the same attributes and the same number of tuples, two sequences of queries

$$\bar{\varphi}_1 = (\varphi_1^{(1)}, \ldots, \varphi_1^{(q)}), \bar{\varphi}_2 = (\varphi_2^{(1)}, \ldots, \varphi_2^{(q)})$$

and presents pairs $(T_1, \bar{\varphi}_1)$, $(T_2, \bar{\varphi}_2)$ to the sender.

3. The sender randomly and uniformly chooses β from set $\{1, 2\}$ and presents $(E_k(T_i), \bar{E^*}_k(\bar{\varphi}_i))$ to the adversary.

4. The adversary tries to guess β.

5. If the adversary guesses β correctly, the outcome of the experiment is 1, and 0 otherwise.

And again, in analogy with Definition 2.5.6, the database privacy homomorphisms is indistinguishably secure if the probability that the adversary can correctly guess bit β in polynomial time cannot be significantly greater than $1/2$.

Formally the experiment can be described as follows:

Experiment Exp_A^{DPH}
$k \xleftarrow{R} \mathcal{K} = \{0,1\}^n$
T_1, T_2 – tables with the same attributes and of the same cardinality
$\bar{\varphi}_1 = (\varphi_1^{(1)}, \ldots, \varphi_1^{(t)}), \bar{\varphi}_2 = (\varphi_2^{(1)}, \ldots, \varphi_2^{(q)}), q = q(n) \leq poly(n)$ – queries
$\beta \xleftarrow{R} \{1, 2\}$
$g \leftarrow A(E_k(T_i), \bar{E}_k^*(\bar{q}_i))$
If $\beta = g$ return 1, else return 0

Definition 3.2.1 (database privacy homomorphism indistinguishable security)**.** *Database privacy homomorphism $(\mathcal{K}, E, E^*, D)$ is indistinguishably secure if for every PPT A, every positive polynomial $p(\cdot)$ the probability of Exp_A^{DPH} returning 1 cannot be significantly greater than the probability of randomly guessing it:*

$$\Pr[Exp_A^{\mathrm{DPH}} = 1] - \frac{1}{2} < \frac{1}{p(n)}$$

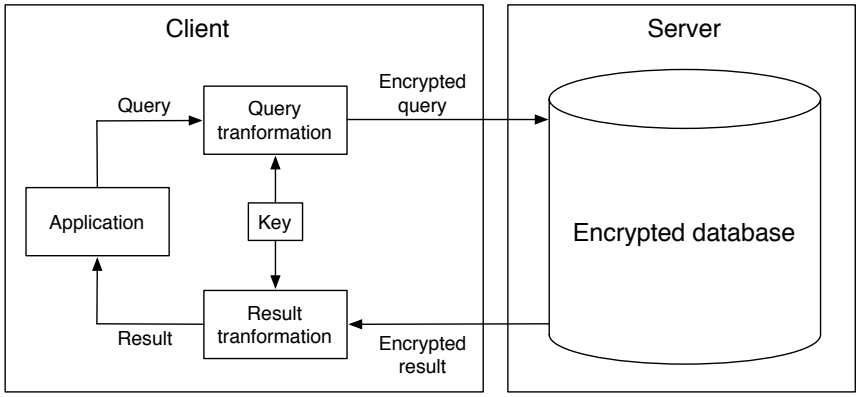


Figure 3.2: Architecture of a securely outsourced database management system

This definition confronts the adversary with a challenge that includes an encrypted table and associated encrypted queries. This reflects what the ASP will gather from the client over time that is a more accurate representation of reality. Figure 3.2 displays an architecture of a system supporting indistinguishably secure database outsourcing.

But how, looking at a database privacy homomorphism, can we tell whether or not it is secure? In order to find necessary conditions for a database privacy homomorphism to be indistinguishably secure, the related concept of cPIR proves helpful. Formally, cPIR can be defined as follows:

Experiment Exp_A^{cPIR}

$k \xleftarrow{R} \mathcal{K} = \{0,1\}^n$

T – table

$\bar{\varphi}_1 = (\varphi_1^{(1)}, \ldots, \varphi_1^{(q)}), \bar{\varphi}_2 = (\varphi_2^{(1)}, \ldots, \varphi_2^{(q)}), q = q(n) \leq poly(n)$ – queries

$\beta \xleftarrow{R} \{1,2\}$

$g \leftarrow A(E_k(T), \bar{E}_k^*(\bar{\varphi}_i))$

If $\beta = g$ return 1 else return 0

Definition 3.2.2 (computationally private information retrieval)**.** *Protocol* $(\mathcal{K}, E, E^*, D)$ *implements cPIR if for every PPT* A*, every positive polynomial* $p(\cdot)$*, all sufficiently large* n *the probability of* Exp_A^{cPIR} *returning 1 cannot be significantly greater than the probability of randomly guessing it:*

$$\Pr[Exp_A^{\text{cPIR}} = 1] - \frac{1}{2} < \frac{1}{p(n)}$$

With cPIR, the necessary conditions for indistinguishable security of a database privacy homomorphism follow.

Theorem 2. *A database privacy homomorphism* $(\mathcal{K}, E, E^*, D)$ *is indistinguishably secure in the sense of Definition 3.2.1 only if*

1. *Encryption scheme* $(\mathcal{K}, E, D)$ *is indistinguishably secure.*

2. *The database privacy homomorphism provides cPIR.*

Proof. For each condition, we construct a direct contradiction to Definition 3.2.1 from assuming its violation.

1. Assume $(\mathcal{K}, E, D)$ is distinguishable, or that there exist such T_1, T_2 that the adversary can distinguish between $E_k(T_1)$ and $E_k(T_2)$ with non-negligible probability. Then the adversary can distinguish between $(E_k(T_1), \bar{E}_k^*(\bar{\varphi}_1))$ and $(E_k(T_2), \bar{E}_k^*(\bar{\varphi}_2))$, even if she simply ignores any $\bar{\varphi}_1$ and $\bar{\varphi}_2$.

2. If the database privacy homomorphism does not provide cPIR, then the adversary can use the query sequences $\bar{\varphi}_1, \bar{\varphi}_2$ from any cPIR attack to distinguish between $(E_k(T), \bar{E}_k^*(\bar{\varphi}_1))$ and $(E_k(T), \bar{E}_k^*(\bar{\varphi}_2))$ with non-negligible probability.

□

These conditions are significantly stronger than the requirements for indistinguishably secure encryptions of the table and queries. There is also a gap between Definition 3.2.1 and Theorem 2: the latter gives a set of *necessary*, not *sufficient* conditions for the former to hold. Even if we find a database privacy homomorphism that satisfies the latter, we have no guarantee that it is secure. However, it can give us insights as to how hard it can be to build such a privacy homomorphism.

The existing cPIR algorithms impose a significant computational and communicative overhead on the client and the ASP. The only algorithm known to the authors that provides both cPIR and hides the queried data is described in [BKOW07]. However, as we have already mentioned, all existing cPIR protocols are still more expensive in terms of communicative overhead compared to the solution in which the client downloads the entire database. Moreover, if a database privacy homomorphism should support queries returning results containing different numbers of tuples, the adversary can compose pairs $(T_1, \bar{\varphi}_1)$, $(T_2, \bar{\varphi}_2)$ such that $\bar{\varphi}_1(T_1) = \emptyset$ and $|\bar{\varphi}_2(T_2)| > 1$. The only way to prohibit the adversary from distinguishing between $(E_k(T_1), \bar{E}_k^*(\bar{\varphi}_1))$

and $(E_k(T_2), \bar{E}^*_k(\bar{\varphi}_2))$ is to conceal the sizes of query results by returning the whole table in response to any `SELECT` query.

Therefore, it may be reasonable to assume that the client wishing to privately outsource her database could consider a trade-off between privacy and functionality. For example, if the client issues only exact selects to the outsourced database, a possible trade-off may be to allow each query to reveal the number tuples in the result while keeping all other information hidden. This is formalized in the following relaxation of Definition 3.2.1:

Definition 3.2.3 (Database privacy homomorphism indistinguishable security up-to-frequency)**.** *A database privacy homomorphism $(\mathcal{K}, E, E^*, D)$ is indistinguishably secure up-to-frequency (of queried elements) if*

1. *The encryption scheme $(\mathcal{K}, E, D)$ is indistinguishably secure.*

2. *For every PPT algorithm A there exists a PPT A' such that for every table T, every sequence of different exact select queries $\bar{\varphi} = (\varphi_1, \ldots, \varphi_q)$ $q = q(n) \leq poly(n)$ and corresponding operations $\bar{\psi} = (\psi_1, \ldots, \psi_q) = (E^*_k(\varphi_1), \ldots, E^*_k(\varphi_q))$, every polynomially bounded functions $f(\cdot), h(\cdot)$, every polynomial $p(\cdot)$ and all sufficiently large n,*

$$\Pr[A(E_k(T), \bar{\psi}, h(T, \bar{\varphi})) = f(T)] < \Pr[A'(E_k(T), \bar{R}_{E_k(T)}(\bar{\psi}), h(T, \bar{\varphi})) = f(T)] + \frac{1}{p(n)}$$

where $\bar{R}_{E_k(T)}(\bar{\psi}) = (R_{E_k(T)}(\psi_1), \ldots, R_{E_k(T)}(\psi_q))$ are sets of references pointing to the tuples of encrypted table $E_k(T)$ that constitute resulting sets of corresponding exact select queries $\bar{\varphi}$.

Definition 3.2.3 suggests a statistical attack exploiting possible dependencies between the information that is legitimately leaked and the information that is supposed to stay secret. By monitoring the encrypted flow of queries and their results, the adversary may be able to gather enough information about the distribution of values of the queried attributes to launch such an attack.

As an example, consider an indistinguishably secure up-to-frequency database privacy homomorphism that allows for exact selects only, and a patient database with statistics for three hospitals. For each patient, we store HIV status and the hospital to which they are assigned:

id	hospital	hiv

Now suppose that the ASP knows the database schema, the number of hospitals, has good estimates of the distribution of patients among hospitals ($0.2, 0.3, 0.5$, resp.) and the percentage of HIV-positive patients (say, 8%). In Definition 3.2.3, this knowledge is captured by function $h(\cdot)$.

The client issues the following queries:

```
SELECT * FROM table WHERE hospital = 1;
SELECT * FROM table WHERE hospital = 2;
SELECT * FROM table WHERE hospital = 3;
SELECT * FROM table WHERE hiv = "positive";
```

Since the sizes of the results are sufficiently different, the ASP can guess the queries with high confidence. This in itself is not a problem: our concern is privacy of the database, not privacy of the queries. But by intersecting the answers to the first and the fourth query, the ASP can now infer the ratio of HIV-positive patients in hospital 1!

In order to find a way to rule out this threat, we can impose validity restrictions on what the client can do with the database. Thus we can easily limit the queries to those that do not reveal anything if the frequencies are revealed. We call those queries *safe*.

Definition 3.2.4 (Safe and unsafe queries)**.** *A safe query with respect to table T is a query that yields always exactly one output tuple if run on T. Unsafe queries are queries that are not safe.*

As an example, consider an RFID reader that reads a unique identification code from an RFID tag attached to some item and extracts information associated with this code (name, price, etc.) from an encrypted outsourced database. Since for every item the code is unique and the reader sends requests only for existing codes, the reader issues only safe queries.

Are there queries that are intuitively safe but are not covered by this definition? If the number of resulting tuples varies freely, the hospital example demonstrates how the ASP can run a successful attack on the encrypted database and infer some sensitive information. Note that if all the queries return exactly k tuples for $k > 1$, assuming tuple-wise encryption of the tables it is possible to craft tables with two attributes a_1, a_2 and queries φ_1, φ_2 such that for one table the queries produce intersecting result sets, and for another they produce non-intersecting result sets. This allows the ASP to infer dependencies between values of different attributes and use them for an attack. For example, if the adversary suspects that attribute a_1 contains names of hospitals and a_2 contains last names of patients, by observing such intersections she can try to estimate the number of family members that were treated in the same hospital. Hence, Definition 3.2.4 is optimal in the sense that any relaxation would include queries that are breaking Definition 3.2.3.

3.3 A Database Privacy Homomorphism Preserving Exact Selects

This section introduces a framework that provides a general approach for building database privacy homomorphisms supporting a practically relevant subset of database operations.

3.3.1 Basic Idea

We now describe an approach for building a class of database privacy homomorphisms that preserve exact selects. The resulting homomorphisms satisfy Definition 3.2.3 and guarantee data privacy if used for scenarios in which the adversary never obtains access to unsafe encrypted queries. This is sufficient in all scenarios described above, in which the client trusts the ASP not to abuse her access to unsafe queries. It is also applicable if the client accesses her data only via exact selects on existing values of the primary key: any such query is safe by definition.

Our approach is based upon the intuitive analogy between running exact selects on a database table and searching a set of documents by a keyword. A table can be seen as a set of documents: each tuple is one document, and the attribute–value pairs of the tuples are the words of the document. We formally establish a structure-preserving mapping from a database table to such document sets. Searchable encryption schemes allow encrypting a set of plaintext documents and then performing word search on encrypted documents without decrypting them and the search word. Descriptions of such schemes can be found in [Goh03; SWP00]. We use [SWP00] to demonstrate our results, but any other scheme can be used as well.

3.3.2 Mapping

In this section, we define a mapping from tables to sets of documents. Given some relational schema R, let $\mathbb{D} = \{a_i : d_{ki}\}$, $d_{ki} \in D_i$ be the set of all attribute-value pairs in R, and let $W = \{w_i\}$ be a set of words (i.e., finite bit strings) such that the total number of words $|W| = |\mathbb{D}| = \sum_{i=1\ldots l} |D_i|$. There is always a bijective mapping from attribute-value pairs to words:

$$\Phi : \mathbb{D} \mapsto W,$$
$$\Phi(a_i : d_{ki}) = w_m$$

For any tuple $\langle a_1 : d_{k1}, \ldots, a_l : d_{kl} \rangle$ there is a corresponding set $V_k = \{\Phi(a_1 : d_{k1}), \Phi(a_2 : d_{k2}), \ldots, \Phi(a_l : d_{kl})\}$ of corresponding words. V_k is called the

document corresponding to tuple k.[2] For every table T (which is a tuple set), there is a corresponding document set $U = \{V_1, \ldots, V_m\}$. Analogous to $\mathbb{R}$, we write $\mathbb{U} = \{V_k\}^s$ for the set of all document sets. Using the bijection Φ, we obtain a mapping $\Lambda : \mathbb{R} \mapsto \mathbb{U}$ of tables to document sets:

$$\begin{aligned} &\Lambda(\{\langle a_1 : d_{m1}, \ldots, a_l : d_{ml}\rangle\}_m) = \\ &\{\{\Phi(a_1 : d_{m1}), \ldots, \Phi(a_l : d_{ml})\}\}_m \end{aligned} \tag{3.1}$$

Note that the mapping $\Lambda : \mathbb{R} \mapsto \mathbb{U}$ is bijective.

3.3.3 Homomorphism

We now show that Λ maps the set U with keyword search to an equivalent schema R with exact selects. In other words, we define a homomorphism that projects keyword searches into exact selects. This is not a database privacy homomorphism, yet because there are no confidentiality considerations involved, we only construct a correspondence between the two In this section. In the next section, we will build encryption around this correspondence. We define keyword search on document sets as follows:

Definition 3.3.1 (Keyword search)**.** *Let* $w \in W$*. Then* $\varsigma_w : \mathbb{U} \mapsto \mathbb{U}$ *is a* keyword search operation *if*

$$\varsigma_w(U) = \{V_k | V_k \in U,\, w \in V_k\}$$

Operation ς_w maps a set of documents to the subset of these documents all of which contain w. For example:

$$\text{if} \quad U = \left\{ \begin{array}{c} \{\text{“ab”,“a”,“bc”}\} \\ \{\text{ “a”,“cba”,“c”}\} \\ \{\text{ “ac”,“ca”,“aa”}\} \end{array} \right\}, \quad \text{then}$$

$$\varsigma_{\text{“ac”}}(U) = \left\{ \begin{array}{c} \{\text{ “ab”,“ac”,“bc”}\} \\ \{\text{ “ac”,“ca”,“aa”}\} \end{array} \right\}$$

Now consider the systems $(\mathbb{R}, \{\sigma_{a_i:d_j}\})$ on the one hand and $(\mathbb{U}, \{\varsigma_{w_{ij}}\})$ on the other. Using the bijection Φ between attribute-value pairs and words, we can define a mapping between the two search operations:

$$\Psi : \{\sigma_{a_i:d_j}\} \mapsto \{\varsigma_{w_{ij}}\},\ \Psi(\sigma_{a_i:d_j}) = \varsigma_{\Phi(a_i:d_j)} \tag{3.2}$$

that defines the pairs of operations preserved by the homomorphism.

[2] One could model documents as sequences and not sets, but sets are strictly more general, and for us the word order is irrelevant.

The mapping Λ is a homomorphism with respect to $(\mathbb{R}, \sigma_{a_i:d_j})$ and $(\mathbb{U}, \varsigma_{\Phi(a_i:d_j)})$: If $T \in \mathbb{R}$, then

$$\Lambda(\sigma_{a_i:d_j}(T)) = \Psi(\sigma_{a_i:d_j})(\Lambda(T))$$

3.3.4 Defining the Database Privacy Homomorphism

The elements of set $\mathbb{U}$ are sets of documents. So, if $U = \{V_1, \ldots, V_m\} \in \mathbb{U}$, then $V_k = \{w_{k1}, \ldots, w_{kl}\} \in U$ represents a document, consisting of the words w_i. Now, search on encrypted data can be favored as a privacy homomorphism: A searchable encryption scheme is a tuple $\Gamma = (\mathcal{K}, E, E^*, D)$, where $E : \mathcal{K} \times \mathbb{U} \mapsto \mathbb{C}$ maps a key $k \in \mathcal{K}$ and a set of documents U into a ciphertext C , $E^* : \mathcal{K} \times \{\varsigma_i\} \mapsto \{E_k^*(\varsigma_i)\}$ maps a key $k \in \mathcal{K}$ and a search query ς to an encrypted query $E_k^*(\varsigma)$, and $D : \mathcal{K} \times \mathbb{C} \mapsto \mathbb{U}$ maps a key k and a ciphertext C back to a plaintext U. (As in the case of database privacy homomorphisms, there is no need for decryption of the queries.)

For any such searchable encryption scheme and the mappings Λ and Ψ introduced above, there is a database privacy homomorphism that preserves exact selects:

Theorem 3. *If $(\mathcal{K}, E, E^*, D)$ is a searchable encryption scheme mapping the plaintexts system $(\mathbb{U}, \{\varsigma_i\})$ to the ciphertext system $(\mathbb{C}, \{E_k^*(\varsigma_i)\})$, then $(\mathcal{K}, E \circ \Lambda, E^* \circ \Psi, \Lambda^{-1} \circ D)$ is a privacy homomorphism mapping the plaintext system $(\mathbb{R}, \{\sigma_{a_i:d_j}\})$ to the ciphertext system $(\mathbb{C}, \{(E_k^* \circ \Psi)(\sigma_{a_i:d_j})\})$.*

Proof. Directly follows from the fact that Λ is bijective and homomorphic, and from Definition 2.6.2. □

Theorem 3 constitutes a method to build a secure database privacy homomorphisms preserving exact selects: If we are given relational schema R and searchable encryption scheme $\Gamma = (\mathcal{K}, E, E^*, D)$, then a database privacy homomorphism for R is built as follows:

1. Choose a set W such that $|W| = |\mathbb{D}|$.
2. Choose a bijective mapping $\Phi : \mathbb{R} \mapsto W$ suitable to Γ.
3. Using W and Φ, generate Λ and Ψ.
4. Output $\Delta = (\mathcal{K}, E \circ \Lambda, E^* \circ \Psi, \Lambda^{-1} \circ D)$.

By Theorem 3, Δ is a privacy homomorphism on R that maps tables T and exact select operations $\sigma_{a_i:d_j}$ to ciphertexts C and $(E_k^* \circ \Psi)(\sigma_{a_i:d_j})$, respectively.

3.3.5 Security Proof

It remains to be shown that the database privacy homomorphism Δ inherits the security characteristics of the searchable encryption scheme Γ, i.e., that it is secure if the client is not submitting any unsafe queries to the ASP.

First, we introduce a definition of security for a searchable encryption scheme that is an analogue of Definition 3.2.3:

Definition 3.3.2 (indistinguishability up-to-frequency for a searchable encryption scheme)**.** *A searchable encryption scheme $\Gamma = (\mathcal{K}, E, E^*, D)$ reveals nothing but the frequencies of searched words if*

1. *Encryption scheme $(\mathcal{K}, E, D)$ is indistinguishably secure.*
2. *For every PPT algorithm A there exists a PPT algorithm A' such that for every sequence of search queries $\bar{\varsigma}$, every set of documents U, every polynomially bounded functions $f(\cdot), h(\cdot)$, every polynomial $p(\cdot)$ and all sufficiently large n,*

$$\Pr\left[A(E_k(U), E_k^*(\bar{\varsigma}), h(U, \bar{\varsigma})) = f(U)\right] < \Pr\left[A'(E_k(U), \bar{N}(\bar{\varsigma}(U)), h(U, \bar{\varsigma})) = f(U)\right] + \frac{1}{p(n)}$$

Then we prove that the proposed database privacy homomorphism Δ is secure in the sense of the introduced security definition:

Theorem 4. *Let $(\mathcal{K}, E, E^*, D)$ be a secure (in the sense of Definition 3.3.2) searchable encryption scheme. Then the database privacy homomorphism $\Delta = (\mathcal{K}, E^* \circ \Psi, E \circ \Lambda, \Lambda^{-1} \circ D)$ as constructed in Theorem 3 is secure (in the sense of Definition 3.2.3).*

Proof. We show that any violation of Definition 3.2.3 for Δ yields a violation of Definition 3.3.2 for Γ.

(1) Suppose $(\mathcal{K}, E \circ \Lambda, \Lambda^{-1} \circ D)$ is not indistinguishably secure, i.e., there exist tables T_1 and T_2 such that the adversary can distinguish between $E_k \circ \Lambda(T_1)$ and $E_k \circ \Lambda(T_2)$ (with non-negligible probability). Then given two encrypted sets of documents $\Lambda(T_1)$, $\Lambda(T_2)$ the adverary can distinguish between $E_k(\Lambda(T_1))$ and $E_k(\Lambda(T_2))$, since $E_k(\Lambda(T_i)) = (E_k \circ \Lambda)(T_i)$. This violates condition 1 of Definition 3.3.2.

(2) Since $(\mathcal{K}, E, E^*, D)$ reveals nothing but the frequencies of searched words, for every probabilistic polynomial-time algorithm A there always exists a probabilistic polynomial-time algorithm A' such that for any set of documents $\Lambda(T)$, any sequence of search queries $\bar{\Psi}(\bar{\varphi})$, and any polynomially

bounded functions $f(\cdot), h(\cdot)$, every polynomial $p(\cdot)$ and all sufficiently large n,

$$\Pr[A(E_k(\Lambda(T)), \bar{E^*}_k(\bar{\Psi}(\bar{\varphi})), h(\Lambda(T), \bar{\Psi}(\bar{\varphi})) = f(\Lambda(T))] <$$
$$\Pr[A'(E_k(\Lambda(T)), \bar{N}(\bar{\Psi}(\bar{\varphi})(\Lambda(T))), h(\Lambda(T), \bar{\Psi}(\bar{\varphi})) = f(\Lambda(T))] + \frac{1}{p(n)}$$

Observe that

$$\begin{aligned}(E_k \circ \Lambda)(T) &= E_k(\Lambda(T)) \\ (\overline{E_k^* \circ \Psi})(\bar{\varphi}) &= \bar{E}_k^*(\bar{\Psi}(\bar{\varphi})) \\ \bar{N}(\bar{\varphi}(T)) &= \bar{N}(\bar{\Psi}(\bar{\varphi})(\Lambda(T))) \\ f'(T) &= f(\Lambda(T)) \\ h'(T, \bar{\varphi}) &= h(\Lambda(T), \Psi(\bar{\varphi}))\end{aligned}$$

Therefore the upper inequality can be rewritten as

$$\Pr\left[A((E_k \circ \Lambda)(T), (\overline{E_k^* \circ \Psi})(\bar{\varphi}), h'(T, \bar{\varphi})) = f'(T)\right] <$$
$$\Pr\left[A'((E_k \circ \Lambda)(T), \bar{N}(\bar{\varphi}(T)), h'(T, \bar{\varphi})) = f'(T)\right] + \frac{1}{p(n)}$$

and condition 2 of Definition 3.2.3 for Δ is satisfied. □

3.3.6 Example

We now deploy the method described in the last section to construct a database privacy homomorphism based on the searchable encryption scheme proposed in [SWP00].

First we define word length and document length as global constants. (There are simple extensions that allow for variable-length words and documents, but for our simple database schema this does not help much, and we skip it here for the sake of clarity. By `'#'` we define the padding symbol.

Consider the following database relation

`Emp(name:string[9],dept:string[5],salary:int)`

The privacy homomorphism is defined as follows:

$$\begin{aligned}W &= \{d_1|\ \texttt{"NM"}, d_2|\ \texttt{"DP"}, d_3|\ \texttt{"SL"} | d_1, d_2, d_3 \in\ \texttt{string[9]}\} \\ \Phi(\texttt{name:}d) &= \texttt{pad}(d)|\ \texttt{"NM"},\ \text{where}\ \ \texttt{pad}(d) \in\ \texttt{string[9]} \\ \Phi(\texttt{dept:}d) &= \texttt{pad}(d)|\ \texttt{####}|\ \texttt{"DP"},\ \text{where}\ \ \texttt{pad}(d) \in\ \texttt{string[5]} \\ \Phi(\texttt{salary:}d) &= \texttt{pad}(\ \texttt{toString}(d))|\ \texttt{#####}|\ \texttt{"SL"},\ \text{where}\ d \in\ \texttt{int}\end{aligned}$$

Then, Λ maps tuples as follows (see (3.1)):

$$V_k = \Lambda(\langle \texttt{name:"Montgomery"}, \texttt{dept:"HR"}, \texttt{salary:7500} \rangle) = \\ \{\Phi(\texttt{name:"Montgomery"}), \Phi(\texttt{dept:"HR"}), \Phi(\texttt{salary:7500})\} = \\ \{\texttt{"MontgomeryNM"}, \texttt{"HR########DP"}, \texttt{"7500######SL"}\}$$

Analogously, other tuples are mapped to sets, which are stored as strings, consisting of 3 words of 12 bytes each. But we need to be careful with the linear representation of sets we chose. In relational algebra, tuple attributes are not ordered. The set $V_k = \{w_{k1}, \ldots, w_{kl}\}$ is not ordered either. However, the resulting document strings fix the order of elements w_{ki} and, hence, the order of the attributes in the tuples. In order to be consistent with the framework described above, words of each string must be permuted randomly and independently of all other document strings, such that the position of a word in a document is uniformly random. Any correlation between word order and attributes would allow the adversary to distinguish queries with different attributes in the equality conditions, and that would break the security of the database privacy homomorphism. Thus, the words set { `"MontgomeryNM"`, `"HR########DP"`, `"7500######SL"`}, will be stored as the string with the order of words defined by a random permutation

`"HR########DPMontgomeryNM7500######SL"`

At this point, the resulting strings are encrypted using the searchable encryption scheme and stored on ASP's server.

Mapping Ψ, as introduced in (3.2), establishes the link between exact select and keyword search. For example, the exact select query $\sigma_{\texttt{name:"Montgomery"}}$ will be mapped to the search operation $\varsigma_{\texttt{"MontgomeryNM"}}$, and processed as a search operation, returning a set of encrypted strings. This set is then decrypted and by applying Λ^{-1} mapped to corresponding tuples producing the result of the issued query.

Following [SWP00], the searchable encryption scheme indistinguishably encrypts sets of documents and reveals nothing but the number of documents sharing the queried word. Thus, according to Theorem 4, the resulting database privacy homomorphism is secure in the sense of Definition 3.2.3.

3.3.7 Boolean Expressions

It is possible to increase the subset of SQL that can be handled by Boolean operations and process exact selects with conjunction and disjunctions, and negations of multiple equality conditions. This is because Boolean operations

in select conditions are really set operations:

$$\sigma_{(a_1:d_1 \vee (a_2:d_2 \wedge a_3:d_3))} = \sigma_{a_1:d_1} \cup (\sigma_{a_2:d_2} \cap \sigma_{a_2:d_2})$$

For every `OR` operation, the query is split in two (one for each operand), and the result is the union of the results of the two queries. `AND` can be implemented similarly with an intersect operation, and `NOT` by subtracting the un-negated result from the entire table. All set operations can be carried out by the server with constant computation overhead. However, the negation can only be implemented for searchable encryption schemes that do not include in the resulting set of a search operation documents (tuples), which do not contain a searchable keyword: i.e., searchable encryption schemes not requiring postfiltering. This is due to the fact that by subtracting the erroneously included tuples the ASP may remove tuples that should belong to the resulting set and, obviously, the postfiltering on the client side cannot help in recovering them. And while due to such limitations the schemes described in [Goh03; SWP00] will not allow using the negation operator in the queries, in Section 3.4 we propose a searchable encryption scheme that guarantees that the probability of such an error is negligible, making the negation operation possible.

3.3.8 Complexity

The performance of the obtained database privacy homomorphisms depends on the time required for processing exact select queries and the time required for encryption and decryption. The plaintext tuples are encrypted once when inserted into the encrypted table. Decryption is performed on the results, which are usually significantly smaller than the queried tables. Therefore, the most time-consuming operation is the processing of the exact select queries. The time required for processing one such query on a table consisting of l columns and m tuples is the same as the time required to perform search on m encrypted documents consisting of l words. Searchable encryption schemes available at present perform such a search in time varying from $O(m)$ to $O(m \cdot l)$, depending on the scheme being used. Therefore, processing an exact select query will also require $O(m)$ to $O(m \cdot l)$ time. Indexing techniques can be applied for reducing the search time. A detailed discussion of the indexing issue is presented in Section 3.4.5.

Searchable encryption schemes (and the one we used as an example particularly) may erroneously return documents (tuples) not containing the keyword. However, the probability of such an error is relatively small; and thus, the erroneous tuples do not create a significant computational overhead. The only related issue is that the client must perform postfiltering of every result.

3.4 New Searchable Encryption Scheme

In this section, we propose a new searchable encryption scheme. Compared to the existing schemes supporting search on encrypted data, our solution reduces the probability of returning incorrect results of a search to a negligible level. Moreover, it allows us to expand the number of possible database operations when applied to the framework we described in the previous section.

3.4.1 Basic Idea

In this section, we propose an encryption scheme that can serve as a database privacy homomorphism for a well-defined subset of database operations. We show how to perform encryption and decryption of a database, discuss the database operations that are preserved by the database privacy homomorphism based on the proposed encryption scheme, prove that it complies with the strongest notion of indistinguishable security – indistinguishable security under chosen-ciphertext attack, and that when performing an exact select, it reveals nothing but the frequency of the queried elements.

The idea behind the scheme is to separately encrypt each attribute value of a database relation, augmenting the resulting ciphertexts with additional pieces of information, viz., a search tags that allow executing a search on the ciphertexts without getting any information about the corresponding plaintext values. To perform a search, only a search tag has to be analyzed, keeping the details about the corresponding plaintext values hidden.

3.4.2 Construction

We build our scheme as the combination of cryptographic primitives. When implementing the encryption scheme as a computer program, the primitives are substituted with their implementations that are *believed* to satisfy necessary security requirements (e.g., DES, RSA, MD5, SHA etc.). By saying "believed," we mean that so far as there were no successful attacks on these implementations. In case a security breach is found, the compromised implementation can be substituted by another construct that possesses the needed properties and is considered to be secure.

Our encryption scheme uses the following cryptographic primitives:

- $(\mathcal{K}, E, D)$, $\mathcal{K} = \{0,1\}^m$, $\mathcal{X} = \{0,1\}^m$, $E : \mathcal{K} \times \mathcal{X} \mapsto \mathcal{C}$ is a symmetric encryption schema that is indistinguishably secure under chosen plaintext attack with identical key and plaintext spaces.

- $(\mathcal{K}^0, E^0, D^0)$, $E^0 : \mathcal{K}^0 \times \mathcal{X} \mapsto \mathcal{C}^0$ is a symmetric encryption schema that is indistinguishably secure under chosen plaintext attack.
- $P : \mathcal{K}' \times \mathcal{X} \mapsto \mathcal{X}$, $\mathcal{X} = \{0,1\}^m$ is a pseudo-random permutation. Since $\mathcal{K} = \mathcal{X}$ we can also write $P : \mathcal{K}' \times \mathcal{X} \mapsto \mathcal{K}$

Key generation. The sender generates the encryption key $\hat{k}$ that is a triple (k_0, k_1, k_2), where $k_0 \overset{\text{R}}{\leftarrow} \mathcal{K}^0$, $k_1 \overset{\text{R}}{\leftarrow} \mathcal{K}'$, $k_2 \overset{\text{R}}{\leftarrow} \mathcal{K}'$: k_0 is the key for encryption scheme $(\mathcal{K}^0, E^0, D^0)$, k_1, k_2 are the keys for pseudo-random permutation P (k_1, k_2 are chosen independently).
Encryption. Suppose that the client (the sender) wants to encrypt a relational database that consists of several relations. Each relation is encrypted separately, so we describe the encryption algorithm for an arbitrary attribute value of relation $R(a_1 : D_1, \ldots, a_l : D_l)$. The encryption algorithm maps the relation R to an encrypted relation R^E that has the same number of attributes but the domains of the attributes are changed to binary strings. Since the information about the domains will be not available after encryption, the client is responsible for saving this information and performing correct type conversions during the decryption process (this will be discussed later in more detail).

Before starting the encryption, the client generates key $\hat{k}$ and then performs tuple-by-tuple encryption of relation R, separately encrypting each attribute value. Let $x \in D_i$ be a plaintext value of attribute a_i. The encryption algorithm treats plaintext x as a binary value and encrypts it by performing the following steps:

1. Plaintext x is encrypted with encryption function E^0 and key k_0: $c = E^0_{k_0}(x)$.
2. Pseudo-random permutation P generates key k_s: $k_s = P_{k_1}(x)$. Key k_s will be used for generating the search tag.
3. Plaintext x is deterministically encrypted by pseudo-random permutation P with key k_2: $s = P_{k_2}(x)$
4. Using ciphertext s and key k_s the search tag is generated: $t = E_{k_s}(s)$.
5. The output of the algorithm is the pair (t, c).

With $\hat{E}$ denoting the encryption algorithm, whole procedure can be described as

$$\hat{E}_{\hat{k}}(x) := (E_{P_{k_1}(x)}(P_{k_2}(x)), E^0_{k_0}(x)), \tag{3.3}$$

Attribute of R	**Attribute of R^E**	**Type of Attribute**
ID	f4FR32	int
Name	aSC3f7	string[100]
	...	
Address	sF3nD4	string[200]

Table 3.1: Corresponding attributes and data types

where $\hat{k} = (k_0, k_1, k_2)$.

After the encryption procedure was applied to each attribute value of tuple $\langle a_1 : x_1, \ldots, a_l : x_l \rangle$, the resulting ciphertexts form a new tuple $\langle a_1^E : (t_1, c_1), \ldots, a_l^E : (t_l, c_l) \rangle$ that belongs to relation R^E. In order to hide the structure of the database, the names of the attributes should be changed ($a_i \neq a_i^E$). To correctly decrypt the encrypted relation, the client should store the information about the correspondences between the attributes of relation R and the attributes of the relation R^E. Also, as mentioned earlier, the encryption changes the domains of the attributes to a raw binary data. In order to be able to convert decrypted values into the corresponding attribute values the information about the domains of original attributes should also be maintained by the client. Table 3.1 illustrates which metadata should be stored by the client in order to be able to correctly decrypt the encrypted tuples.

In order to use the described encryption scheme for encrypting values of different attributes, the domains of relation R^E should be of the same length. This means that before being encrypted, the values should be padded up to the length of the domain that has the longest binary representation. Attributes containing very long values might be split into several shorter attributes. Note, however, that it is very unlikely that values of such attributes values will be used by an exact select (e.g., attributes that contain full address, long text, multimedia data, etc.). In case no select queries are expected for them, they can be encrypted with a conventional indistinguishably secure encryption scheme.

Decryption. The decryption is performed by decrypting the attribute values of every tuple of relation R^E and filling relation R with the corresponding plaintexts tuples taking into account the information from Table 3.1. The decryption of ciphertext (t, c) is performed in a straightforward manner:

$$\hat{D}_{\hat{k}}(t, c) := D_{k_0}(c) = x \tag{3.4}$$

where $\hat{k} = (k_0, k_1, k_2)$.

Using the information stored in Table 3.1 the plaintext is converted to

the appropriate type and saved as the value of the corresponding attribute.

The final scheme is defined as $(\hat{\mathcal{K}}, \hat{E}, \hat{D})$, where $\hat{E}$ is defined according to (3.3), $\hat{D}$ is defined according (3.4) and $\hat{\mathcal{K}} = (\mathcal{K}^0 \times \mathcal{K}' \times \mathcal{K}')$.

3.4.3 Operations on Encrypted Relational Databases

In this section, we discuss the relational operations that are feasible under the proposed scheme together with implications that may arise when some of operations are performed.

The encryption schema described above allows for performing the following subset of relational operations on encrypted relations: exact select, projection and Cartesian product. The scheme also allows for performing union with duplicates, exact update, exact delete and insert.

Exact Select. The proposed encryption scheme allows for performing exact selects[3] on the encrypted relation without decrypting it. Exact selects with more than one selection conditions connected by AND or OR are discussed at the end of this section.

Suppose, that exact select $\sigma_{a_i.x_q}$ should be performed on relation R that is encrypted and stored as R^E. Then the following actions should be performed:

1. The client transforms the query $\sigma_{a_i.x_q}$ into the following triple

$$(q, k_q, a_i^E) = (P_{k_2}(x_q), P_{k_1}(x_q), a_i^E) \tag{3.5}$$

   where a_i^E is the name of the attribute of relation R^E that corresponds to attribute a_i. The corresponding attributes are taken from the structure analogous to Table 3.1.

2. Tuple by tuple, the server checks every value (t, c) of attribute a_i^E for the following equality:

$$D_{k_q}(t) = q \tag{3.6}$$

   The tuples that satisfy the equality are marked.

3. After all the tuples of the relation R^E are checked, the server sends the marked tuples to the client. The search tags of the attribute values are not needed for the decryption and can thus be discarded. That would by a half reduce the amounts of data transferred to the client.

4. Using key k_0, the client decrypts the received ciphertexts.

[3] *Exact selects* are queries that in SQL language can be expressed as `SELECT... FROM...WHERE` ⟨`attribute_name`⟩=⟨`value`⟩.

Recall that, when encrypting plaintext x, the encryption algorithm $\hat{E}$ generates a key $k_s = P_{k_1}(x)$ and a ciphertext $s = E_{k_s}(P_{k_2}(x))$. If the ciphertext (t, c), whose search tag was checked at step 2, is the encryption of x_q, then $k_s = k_q$, $s = q$, and equality (3.6) holds true due to

$$D_{k_q}(t) = D_{k_q}(E_{k_s}(s)) = D_{k_q}(E_{k_s}(P_{k_2}(x_q))) = P_{k_2}(x_q) = q$$

Therefore, all the tuples that have encryption of x_q as the value of attribute a_i^E will be marked and included in the result set.

Note that the triple provided by client does not contain any plaintext values. That allows the server to perform search for $a_i.x_q$ without $a_i.x_q$ being revealed.

However, we cannot call this scheme privacy homomorphism in a strict sense, since the set of marked tuples may contain tuples that do not belong to the actual solution. This can happen due to the following collision:

$$D_{P_{k_1}(x_q)}(E_{P_{k_1}(x)}(P_{k_2}(x))) = P_{k_2}(x_q) \tag{3.7}$$

where $x_q \neq x$, $\hat{k} = (k_0, k_1, k_2)$.

In general the probabilities of such collisions vary depending on encryption scheme $(\mathcal{K}, E, D)$. A good candidate to minimize this probability is the indistinguishably secure under chosen-plaintext attack one-time pad based encryption scheme constructed as follows:

- Key generation: $k \xleftarrow{R} \mathcal{K}$.
- Encryption: $E_k(x) := (r, f_k(r) \oplus x)$, where $f : \mathcal{K} \times \mathcal{X} \mapsto \mathcal{X}$ is a pseudo-random function, $r \xleftarrow{R} \mathcal{X}$.
- Decryption: $D_k(r, c) := f_k(r) \oplus c$.

The scheme is simple, efficient and, according to [Gol04], indistinguishably secure under chosen-plaintext attack.

In order to use this scheme as $(\mathcal{K}, E, D)$ we require $k, r, x \in \{0,1\}^m$ and $f : \{0,1\}^m \times \{0,1\}^m \mapsto \{0,1\}^m$. Using this implementation of $(\mathcal{K}, E, D)$ we can rewrite (3.7) as

$$\begin{gathered} f_{P_{k_1}(x_q)}(r) \oplus f_{P_{k_1}(x)}(r) \oplus P_{k_2}(x) = P_{k_2}(x_q) \\ \Updownarrow \\ f_{P_{k_1}(x_q)}(r) \oplus f_{P_{k_1}(x)}(r) = P_{k_2}(x_q) \oplus P_{k_2}(x). \end{gathered}$$

Consider the ideal case where instead of pseudo-random functions $f_{P_{k_1}(x_q)}$, $f_{P_{k_1}(x)}$ random functions Φ_{k_1}, Φ_{k_2} are used. Then

$$\Pr[\Phi_{k_1}(r) \oplus \Phi_{k_2}(r) = P_{k_2}(x_q) \oplus P_{k_2}(x), x \neq x_q, \, r \xleftarrow{R} \mathcal{X}] = \frac{1}{2^m}$$

The probability that the collision (3.7) will not occur is the probability of the inverse event or

$$\Pr[\Phi_{k_1}(r) \oplus \Phi_{k_2}(r) \neq P_{k_2}(x_q) \oplus P_{k_2}(x), \, x \neq x_q, r \xleftarrow{R} \mathcal{X}] = 1 - 2^{-m}$$

In order to estimate the probability that there will be no collisions when equality (3.6) is checked for a set of different values $\{x_1, \ldots, x_{t(m)}\}$, where $x_i \neq x_q$, $x_i \neq x_j$, $i \neq j$ and $t(m) \leq poly(m)$, we note that in the ideal case, for each x_i the random function Φ_{k_1} is chosen independently and thus the events that correspond to the collisions for each x_i are also independent. Therefore the probability that, when performing an exact select σ_{x_q} on values $\{x_1, \ldots, x_{t(m)})\}$, no collisions occur is

$$(1 - \Pr[\phi(r) \oplus \Phi_{k_2}(r) = P_{k_2}(x_q) \oplus P_{k_2}(x), x \neq x_q, \, r \xleftarrow{R} \mathcal{X}])^{t(m)} = (1 - 2^{-m})^{t(m)}$$

Analogously, for each new query there is a corresponding randomly chosen function Φ_{k_2}. The probability of event $\Re$ that corresponds to the absence of collisions when querying $t(m)$ values with $s(m)$ different queries is

$$\begin{aligned}\Pr(\Re) &= (1 - \Pr[\Phi_{k_1}(r) \oplus \Phi_{k_2}(r) = P_{k_2}(x_q) \oplus P_{k_2}(x), x \neq x_q, r \xleftarrow{R} \mathcal{X}])^{t(m)s(m)} \\ &= (1 - 2^{-m})^{t(m)s(n)}\end{aligned}$$

The lower bound of probability $\Pr(\Re)$ can be estimated as

$$\Pr(\Re) = \left(1 - \frac{1}{2^m}\right)^{t(m)s(m)} > \left(1 - \frac{t(m)s(m)}{2^m}\right) = \left(1 - \frac{p(m)}{2^m}\right) > \left(1 - \frac{1}{p(m)}\right) \tag{3.8}$$

for sufficiently large m and positive polynomial $p(\cdot)$. The probability that there will be at least one collision is $1 - \Pr(\Re) < 1/p(m)$, which is negligible.

This estimation was performed for the case in which functions Φ_{k_1}, Φ_{k_2} are chosen from the family of random functions. If instead we use pseudo-random functions $f_{P_{k_1}(x_q)}, f_{P_{k_1}(x)}$, it can be shown that the probability of the collision will remain negligible.

To do it, first we prove that pairs $(f_{P_{k_1}(x_q)}, f_{P_{k_1}(x)})$ and (Φ_{k_1}, Φ_{k_2}) are indistinguishable:

Lemma 1. *If $f : \mathcal{K} \times \mathcal{X} \mapsto \mathcal{Y}$ is a pseudo-random function and $\Phi : \mathcal{K} \times \mathcal{X} \mapsto \mathcal{Y}$ is a random function, then $(f_{k_1}(x), f_{k_2}(x))$ and $(\Phi_{k_1}(x), \Phi_{k_2}(x))$ are indistinguishable:*

Proof. We have to show that for randomly and uniformly chosen $(k_1, k_2) \stackrel{R}{\leftarrow} \mathcal{K} \times \mathcal{K}$, every PPT oracle algorithm A, every positive polynomial $p(\cdot)$ and all sufficiently large n,

$$\mathrm{Adv}A = |\Pr\left[A^{(f_{k_1}, f_{k_2})} = 1\right] - \Pr\left[A^{(\Phi_{k_1}, \Phi_{k_2})} = 1\right]| < \frac{1}{p(n)}$$

Adv$A$ can be rewritten as

$$\begin{aligned}\mathrm{Adv}A \leq |\Pr\left[A^{(f_{k_1}, f_{k_2})} = 1\right] - \Pr\left[A^{(\Phi_{k_1}, f_{k_2})} = 1\right]| + \\ |\Pr\left[A^{(\Phi_{k_1}, f_{k_2})} = 1\right] - \Pr\left[A^{(\Phi_{k_1}, \Phi_{k_2})} = 1\right]| = \\ \mathrm{Adv}\,1 + \mathrm{Adv}\,2\end{aligned}$$

Let assume that Adv 1 is significant. Then PPT algorithm A can be used as a subroutine for building the following PPT oracle algorithm B that will be able to distinguish between F_k and Φ_k with significant probability:

$$\begin{aligned}&\text{Algorithm } B^{O_{k_1}} : \\ &1 : k_2 \stackrel{R}{\leftarrow} \mathcal{K} \\ &2 : g \leftarrow A^{(O_{k_1}, f_{k_2})} \\ &3 : \text{return } g\end{aligned}$$

Access to oracle $(O_{k_1}(\cdot), f_{k_2}(\cdot))$ for algorithm A is provided by algorithm B via its own oracle $O_{k_1}(\cdot)$ and pseudo-random function $f_{k_2}(\cdot)$, where key k_2 was generated on step 2.

Then the advantage for algorithm B in distinguishing between $f_{k_1}(\cdot)$ and $\Phi_{k_1}(\cdot)$ can be estimated as follows:

$$\begin{aligned}\mathrm{Adv}B \leq \\ |\Pr\left[B^{\Phi_{k_1}} = 1\right] - \Pr\left[B^{f_{k_1}} = 1\right]| = \\ |\Pr\left[A^{(\Phi_{k_1}, f_{k_2})} = 1\right] - \Pr\left[A^{(f_{k_1}, f_{k_2})} = 1\right]| = \\ \mathrm{Adv}\,1 \geq \frac{1}{p(n)}\end{aligned}$$

where $p(\cdot)$ is a positive polynomial. That contradicts to the pseudo-randomness of function $f_k(\cdot)$.

The same argument can be applied to Adv 2. As result we have shown that $(f_{k_1}(x), f_{k_2}(x))$ is indistinguishable from $(\Phi_{k_1}(x), f_{k_2}(x))$ and $(\Phi_{k_1}(x), f_{k_2}(x))$ is indistinguishable from $(\Phi_{k_1}(x), \Phi_{k_2}(x))$:

$$\forall i,\ \mathrm{Adv}\, i < \frac{1}{2 \cdot p(n)},\ i \in \{1,2\} \Rightarrow \mathrm{Adv} A < \frac{1}{p(n)}$$

Therefore $(f_{k_1}(x), f_{k_2}(x))$ is indistinguishable from $(\Phi_{k_1}(x), \Phi_{k_2}(x))$. □

Suppose now that there exist a set of values and a set of queries, such that probability $1 - \Pr(\Re)$ is non-negligible. Then, using these sets, we can build an algorithm that distinguishes between (Φ_{k_1}, Φ_{k_2}) and $(f_{P_{k_1}(x_q)}, f_{P_{k_1}(x)})$ with non-negligible probability. Due to the polynomial sizes of the sets, the distinguishing algorithm will be polynomial time. That will contradict to the indistinguishability of (Φ_{k_1}, Φ_{k_2}) and $(f_{P_{k_1}(x_q)}, f_{P_{k_1}(x)})$. Therefore, the probability of collision in the non-ideal case is also negligible. That means that with sufficient key lengths in most of the cases, queries results will not contain any erroneous tuples. However, since the possibility of an error is not excluded, in some applications the client still may have to recheck the result set in order to ensure its correctness.

To get a feeling on what this could mean for practical applications consider the following example: if the encryption scheme $(\mathcal{K}, E, D)$ uses the random function Φ_k, $m = 128$, and $t(m)s(m) = 10^{20}$ then according to (3.8) $\Pr(\Re) > 1 - 2.9 \cdot 10^{-19}$. That means that if the client issues 10^{10} different exact selects $\sigma_{a.x_i}$, $i = 1, .., 10^{10}$, $x_i \neq x_j$, $i \neq j$, and the attribute a of the queried relation contains 10^{10} different values, the probability that no erroneous tuples will be included in results any of the 10^{10} queries is at least $1 - 2.9 \cdot 10^{-19}$. When instead of the random function a cryptographic primitive is used (e.g., HMAC-SHA-2 [KBC97]), the actual probability might become lower, but still the presented technique may serve as a good approach for estimating the chances of having in the result set a tuple that does not satisfy query conditions.

In order to process an exact select $\sigma_{a_i.x}$ for a relation consisting of u tuples, the server should only check whether equality (3.6) holds true for the value of attribute a_i of every tuple. Every check requires $O(1)$ operations and therefore processing of the query for the whole relation will be done in $O(u)$ operations.

Projection. Since the attributes of the relation are encrypted separately, in order to perform projection $\pi_{a_i,\ldots a_j}(a_1, \ldots a_l)$, the client simply provides the name of the corresponding encrypted attributes and the server performs $\pi_{a_i^E,\ldots a_j^E}(a_1^E, \ldots a_l^E)$ on the encrypted relation.

Cartesian product. Cartesian product of two encrypted relations is carried out just as with unencrypted relations - by returning all combinations of tuples of the encrypted relations. Again, this is possible because the attributes are encrypted separately and, as a result, ciphertexts can be concatenated.
Union with duplicates. The union of two encrypted relations is performed by simply including the tuples of both relations in the resulting one. However, the possible duplicated tuples cannot be removed from the resulting table since the server has no means to determine whether two ciphertexts correspond to identical or different plaintexts.
Exact update. Exact update is feasible due to the feasibility of exact select and separate encryption of the attribute values: exact select allows for specifying the tuples that should be updated and separate encryption allows for replacing the encrypted attribute values of the tuples with the new ones. For example, consider the following update query: `UPDATE table1 SET salary = 3500 WHERE name = "John Smith"`. The client transforms the query into tuple $(c, a_c^E, s, k_s, a_i^E)$ and sends it to the server. The last three values allow running the exact select query for obtaining the tuples to be updated. The first two values are the encryption of the new attribute value (3500) and the attribute name of the encrypted relation that corresponds to the one that should be updated (salary).
Exact delete. In order to run exact delete, the client sends to the server a triple (s, k_s, a_i^E), so that the server can find the tuples to be deleted and then remove them from the encrypted relation.
Insert. To insert a tuple, the client encrypts it and sends to the server. The server appends the arrived encrypted tuple to the corresponding relation.
Logical operations. It is also possible to run operations with conditions consisting of several equalities connected by AND or OR. In case of a pair of equalities connected by a logical operation α, the client sends a pair of triples connected by α to the server: $(s_i, k^{s_i}, a_i^E)\,\alpha\,(s_j, k^{s_j}, a_j^E)$, where $\alpha \in \{\text{AND}, \text{OR}\}$. If $\alpha = \text{AND}$, the server marks the tuple when (3.6) holds true for both triples. If $\alpha = \text{OR}$, the server marks the tuple when (3.6) holds true for one of the triples (conditions built of more than two equalities connected by AND or OR can be treated in an analogous manner).

When there is a negation of the equality condition (NOT operation), the server marks those tuples for which (3.6) does not hold. Note, however, that due to the possibility of having tuples erroneously included in the result of the corresponding exact select, NOT operation (as we discussed in Section 3.3.7) can cause valid tuples to be excluded from the final resulting set. But concerning this scheme the probability of such an error is negligible.

3.4.4 Security Analysis

Indistinguishability under chosen-plaintext attack

Theorem 5. *Encryption scheme $(\hat{\mathcal{K}}, \hat{E}, \hat{D})$ is indistinguishably secure under chosen-plaintext attack.*

First we prove several lemmas. Note that the notation used for defining cryptographic primitives within the lemmas is not related to the cryptographic primitives that are used for building encryption scheme $(\hat{\mathcal{K}}, \hat{E}, \hat{D})$.

Let $(\mathcal{K}, E, D)$, where $\mathcal{K} = \{0,1\}^m$, $\mathcal{X} = \{0,1\}^m$, $E : \mathcal{K} \times \mathcal{X} \mapsto \mathcal{C}$, be indistinguishably secure under chosen-plaintext attack encryption scheme and $F : \mathcal{K}' \times \mathcal{X} \mapsto \mathcal{K}$, where $\mathcal{K}' = \{0,1\}^n$, be a pseudo-random function.

Lemma 2. *The scheme $(\mathcal{K}', E')$, where $E'_{k'}(x) := E_{F_{k'}(x)}(x)$, is indistinguishably secure under chosen-plaintext attack.*[4]

Proof. The indistinguishable security under chosen-plaintext attack of scheme $(\mathcal{K}', E')$ means that for every pair of plaintexts x, y, every PPT algorithm A with access to the encryption oracle, every positive polynomial $p(\cdot)$ and for all sufficiently large n, the advantage

$$\mathrm{Adv}A_{xy}^{E'_{k'}} = |\Pr\left[A^{E'_{k'}}(E'_{k'}(x)) = 1\right] - \Pr\left[A^{E'_{k'}}(E'_{k'}(y)) = 1\right]| < \frac{1}{p(n)}$$

For every algorithm A, we can define an oracle algorithm B that is identical to A except for the steps where A sends x as a query for its oracle: after querying its oracle $O(\cdot)$ with x B also calculates $E_{O(x)}(x)$. Then, defining by $E_{F_{k'}}$ oracle $E'_{k'}(\cdot) = E_{F_{k'}(\cdot)}(\cdot)$, and by $E_{\Phi_{k'}}$ oracle $E_{\Phi_{k'}(\cdot)}(\cdot)$, where $\Phi_{k'}$ is a random function, the latter inequality can be rewritten as

$$\begin{aligned}
\mathrm{Adv}A_{xy}^{E'_{k'}} \leq\ & |\Pr\left[B^{F_{k'}}(E_{F_{k'}(x)}(x)) = 1\right] - \Pr\left[B^{\Phi_{k'}}(E_{\Phi_{k'}(x)}(x)) = 1\right]| + \\
& |\Pr\left[A^{E_{\Phi_{k'}}}(E_{\Phi_{k'}(x)}(x)) = 1\right] - \Pr\left[A^{E_{\Phi_{k'}}}(E_{\Phi_{k'}(x)}(y)) = 1\right]| + \\
& |\Pr\left[B^{\Phi_{k'}}(E_{\Phi_{k'}(x)}(y)) = 1\right] - \Pr\left[B^{\Phi_{k'}}(E_{\Phi_{k'}(y)}(y)) = 1\right]| + \\
& |\Pr\left[A^{E_{\Phi_{k'}}}(E_{\Phi_{k'}(y)}(y)) = 1\right] - \Pr\left[A^{E_{F_{k'}}}(E_{F_{k'}(y)}(y)) = 1\right]| = \\
& \mathrm{Adv1} + \mathrm{Adv2} + \mathrm{Adv3} + \mathrm{Adv4}
\end{aligned}$$

Let assume that Adv1 is significant. Then PPT algorithm B can be used as a subroutine for building the following PPT oracle algorithm C that will

[4]The scheme $(\mathcal{K}', E')$ is not an encryption scheme since the construction of E' does not suppose decryption. Since the decryption is not mentioned in the definitions of indistinguishable security, we can also apply them to such schemes.

be able to distinguish between F_k and Φ_k with significant probability:

$$\begin{aligned}&\text{Algorithm } C^{O_k}:\\&1: g \leftarrow B^{O_k}(E_{O_k(x)}(x))\\&2: \text{return } g\end{aligned}$$

Whenever B needs to query $O_k(\cdot)$, C provides it with the corresponding result using its own oracle.

Then the advantage for algorithm C in distinguishing between $E_{F_{k'}(x)}(x)$ and $E_{\Phi_{k'}(x)}(x)$ can be estimated as follows:

$$\begin{aligned}&\text{Adv}C \leq\\&|\Pr[C^{F_{k'}}=1]-\Pr[C^{\Phi_{k'}}=1]| =\\&|\Pr[B^{F_{k'}}(E_{F_{k'}(x)}(x))=1]-\Pr[B^{\Phi_{k'}}(E_{\Phi_{k'}(x)}(x))=1]| =\\&\text{Adv1} \geq \frac{1}{p(n)}\end{aligned}$$

where $p(\cdot)$ is a positive polynomial. That contradicts to the indistinguishable security under chosen-plaintext attack of scheme $(\mathcal{K}^1, E^1, D^1)$.

The negligibility of Adv2 straightforwardly follows from the indistinguishability under chosen-plaintext attack of encryption scheme $(\mathcal{K}, E, D)$. And the negligibility of Adv3 follows from the fact that $\Phi_{k'}(x)$ and $\Phi_{k'}(y)$ are randomly and uniformly chosen values.

Let assume that Adv4 is significant. Then PPT algorithm A can be used as a subroutine for building the following PPT oracle algorithm D that will be able to distinguish between $F_{k'}$ and $\Phi_{k'}$ with significant probability:

$$\begin{aligned}&\text{Algorithm } D^{O_{k'}}:\\&1: g \leftarrow A^{E_{O_{k'}}}(E_{O_{k'}(y)}(y))\\&2: \text{return } g\end{aligned}$$

Whenever A needs to query its oracle $E_{O_{k'}}(\cdot)$, D provides it with the corresponding result using its own oracle and publicly available encryption algorithm E.

Then the advantage for algorithm D in distinguishing between $F_{k'}$ and $\Phi_{k'}$ can be estimated as follows:

$$\begin{aligned}&\text{Adv}D \leq\\&|\Pr[D^{F_{k'}}=1]-\Pr[D^{\Phi_{k'}}=1]| =\\&|\Pr[A^{E_{\Phi_{k'}}}(E_{\Phi_{k'}(y)}(y))=1]-\Pr[A^{E_{\Phi_{k'}}}(E_{\Phi_{k'}(y)}(y))=1]| =\\&\text{Adv4} \geq \frac{1}{p(n)}\end{aligned}$$

where $p(\cdot)$ is a positive polynomial. That, obviously, contradicts to the pseudo-randomness of $F_{k'}$

Thus, we have shown that under chosen-plaintext attacks $E_{F_{k'}(x)}(x)$ is indistinguishable from $E_{\Phi_{k'}(x)}(x)$, $E_{\Phi_{k'}(x)}(x)$ is indistinguishable from $E_{\Phi_{k'}(x)}(y)$, $E_{\Phi_{k'}(x)}(y)$ is indistinguishable from $E_{\Phi_{k'}(y)}(y)$, and $E_{\phi(y)}(y)$ is indistinguishable from $E_{F_{k'}(y)}(y)$:

$$\forall i,\ \mathrm{Adv}\, i < \frac{1}{4 \cdot p(n)},\ i \in \{1,2,3,4\} \Rightarrow \mathrm{Adv} A^{E'}_{xy} < \frac{1}{p(n)}$$

Therefore $(\mathcal{K}', E')$ is indistinguishably secure under chosen-plaintext attack. □

In the next lemma we prove indistinguishably security under chosen-plaintext attack of the scheme analogous to the scheme considered in Lemma 2, with the pseudo-random function F being substituted by the pseudo-random permutation $P : \mathcal{K}^p \times \mathcal{X} \mapsto \mathcal{X}$.

Lemma 3. *If the encryption scheme $(\mathcal{K}, E, D)$ where $\mathcal{K} = \{0,1\}^m$, $\mathcal{X} = \{0,1\}^m$, $E : \mathcal{K} \times \mathcal{X} \mapsto \mathcal{Y}$ is indistinguishably secure under chosen-plaintext attack, then the scheme $(\mathcal{K}^p, E^p)$, where $\mathcal{K}^p = \{0,1\}^n$, $E^p_{k_p}(x) = E_{P_{k_p}(x)}(x)$ and $P : \mathcal{K}^p \times \mathcal{X} \mapsto \mathcal{K}$ is a pseudo-random permutation, is indistinguishably secure under chosen-plaintext attack.*

Proof. Indistinguishable security under chosen-plaintext attack of scheme $(\mathcal{K}^p, E^p)$ follows from Lemma 2 and the fact that a pseudo-random function and a pseudo-random permutation with the same key and argument spaces are indistinguishable [Gol04]. Otherwise, if assume that the new scheme is not indistinguishably secure under chosen-plaintext attack, it can be shown that there exists a PPT oracle algorithm that is able to distinguish between pseudo-random function F and pseudo-random permutation P. □

Lemma 4. *If the encryption scheme $(\mathcal{K}, E, D)$ where $\mathcal{K} = \mathcal{X}$, $E : \mathcal{K} \times \mathcal{X} \mapsto \mathcal{Y}$ is indistinguishably secure under chosen-plaintext attack, then the scheme $(\mathcal{K}^{pp}, E^{pp})$, where $\mathcal{K}^{pp} = \mathcal{K}^p \times \mathcal{K}^p$, $E^{pp}_{k_{pp}}(x) := E_{P_{k_1}(x)}(P_{k_2}(x))$, $k_{pp} = (k_1, k_2)$, $k_1, k_2 \in \mathcal{K}^p$ and $P : \mathcal{K}^p \times \mathcal{X} \mapsto \mathcal{K}$ is a pseudo-random permutation, is indistinguishably secure under chosen-plaintext attack.*

Proof. Suppose that the scheme $(\mathcal{K}^{pp}, E^{pp})$ is not indistinguishably secure under chosen-plaintext attack. Then there exist plaintexts x, y and a PPT algorithm A that can distinguish between $E^{pp}_{k_1,k_2}(x)$ and $E^{pp}_{k_1,k_2}(y)$ with non-negligible probability. Considering that $P'_{k_1,k_2} := P_{k_1} \circ P^{-1}_{k_2}$ is also a pseudo-random permutation, it can be shown that algorithm A applied to the scheme

$(\mathcal{K}^p, E^p)$ from Lemma 3, where $E^p_{k_1,k_2}(x) := E_{P'_{k_1,k_2}(x)}(x)$, can distinguish between $E^p_{k_1,k_2}(P_{k_2}(x))$ and $E^p_{k_1,k_2}(P_{k_2}(y))$ with non-negligible probability, since $E^p_{k_1,k_2}(P_{k_2}(x)) = E_{P_{k_1}(x)}(P_{k_2}(x))$ and $E^p_{k_1,k_2}(P_{k_2}(y)) = E^p_{k_1}(P_{k_2}(y))$. □

Lemma 5. *If the encryptions schemes* $(\mathcal{K}^1, E^1, D^1)$ *and* $(\mathcal{K}^2, E^2, D^2)$ *are indistinguishably secure under chosen plaintext attack, then the encryption scheme* $(\mathcal{K}^0, E^0, D^0)$ *where* $\mathcal{K}^0 = \mathcal{K}^1 \times \mathcal{K}^2 = \{0,1\}^n$ *and* $E^0_{k_0}(x) := (E^1_{k_1}(x), E^2_{k_2}(x))$, $k_0 = (k_1, k_2)$ *and* $k_1 \xleftarrow{R} \mathcal{K}^1$, $k_2 \xleftarrow{R} \mathcal{K}^2$ *is also indistinguishably secure under chosen-plaintext attack.*

Proof. Indistinguishably security under chosen-plaintext attack of scheme $(\mathcal{K}^0, E^0, D^0)$ means that for every x, y, every PPT algorithms A with access to the encryption oracle, every positive polynomial $p(\cdot)$ and for all sufficiently large n, advantage $\mathrm{Adv}A^{E^0}_{xy} < 1/p(n)$:

$$\begin{gathered}
\mathrm{Adv}A^{E^0}_{xy} \leq \\
|\Pr\left[A^{E^0_{k_0}}(E^1_{k_1}(x), E^2_{k_2}(x)) = 1\right] - \Pr\left[A^{E^0_{k_0}}(E^1_{k_1}(y), E^2_{k_2}(y)) = 1\right]| \leq \\
|\Pr\left[A^{E^0_{k_0}}(E^1_{k_1}(x), E^2_{k_2}(x)) = 1\right] - \Pr\left[A^{E^0_{k_0}}(E^1_{k_1}(y), E^2_{k_2}(x)) = 1\right]| + \\
|\Pr\left[A^{E^0_{k_0}}(E^1_{k_1}(y), E^2_{k_2}(x)) = 1\right] - \Pr\left[A^{E^0_{k_0}}(E^1_{k_1}(y), E^2_{k_2}(y)) = 1\right]| = \\
\mathrm{Adv1} + \mathrm{Adv2}
\end{gathered}$$

Let assume that Adv1 is significant. Then PPT algorithm A can be used as a subroutine for building the following PPT oracle algorithm B:

$$\begin{gathered}
\text{Algorithm } B^{E^1_{k_1}}(\alpha) : \\
1 : k_2 \xleftarrow{R} \mathcal{K}_2 \\
2 : g \leftarrow A^{E^0_{k_0}}(E^1_{k_1}(\alpha), E^2_{k_2}(\alpha)) \\
3 : \text{return } g
\end{gathered}$$

Access to oracle $E^0_{k_0}(\cdot) = (E^1_{k_1}(\cdot), E^2_{k_2}(\cdot))$ for algorithm A will be provided by algorithm B via its own oracle $E^1_{k_1}(\cdot)$ and encryption algorithm E^2 called with key k_2 generated on the step 1 of algorithm B.

Then the advantage for algorithm B in distinguishing between $E^1_{k_1}(x)$ and $E^1_{k_1}(y)$ can be estimated as follows:

$$\begin{gathered}
\mathrm{Adv}B^{E^1}_{xy} \leq \\
|\Pr\left[B^{E^1_{k_1}}(E^1_{k_1}(x)) = 1\right] - \Pr\left[B^{E^1_{k_1}}(E^1_{k_1}(y)) = 1\right]| = \\
|\Pr\left[A^{E^0_{k_0}}(E^1_{k_1}(y), E^2_{k_2}(x)) = 1\right] - \Pr\left[A^{E^0_{k_0}}(E^1_{k_1}(y), E^2_{k_2}(y)) = 1\right]| = \\
\mathrm{Adv1} \geq \frac{1}{p(n)}
\end{gathered}$$

where $p(\cdot)$ is a positive polynomial. That, obviously, contradicts to the indistinguishable security under chosen-plaintext attack of encryption scheme $(\mathcal{K}^1, E^1, D^1)$.

The same argument can be applied to Adv2. As result we have shown that under chosen-plaintext attacks $(E^1_{k_1}(x), E^2_{k_2}(x))$ is indistinguishable from $(E^1_{k_1}(y), E^2_{k_2}(x))$ and $(E^1_{k_1}(y), E^2_{k_2}(x))$ is indistinguishable from $(E^1_{k_1}(y), E^2_{k_2}(y))$:

$$\forall i,\ \mathrm{Adv}\, i < \frac{1}{2 \cdot p(n)},\ i \in \{1, 2\} \Rightarrow \mathrm{Adv} A^{E^0}_{xy} < \frac{1}{p(n)}$$

Therefore $(\mathcal{K}^0, E^0, D^0)$ is indistinguishably secure under chosen-plaintext attack. □

Now we are ready to prove Theorem 5.

Proof. Indistinguishable under chosen-plaintext attack security of encryption scheme $(\hat{\mathcal{K}}, \hat{E}, \hat{D})$ follows from the indistinguishable under chosen-plaintext attack security of encryption scheme $(\mathcal{K}, E, D)$, Lemma 4 and Lemma 5. □

Indistinguishability under Chosen-Ciphertext Attack

Although though the encryption scheme $(\hat{\mathcal{K}}, \hat{E}, \hat{D})$ is indistinguishably secure under chosen-plaintext attack, it is straightforward to show that it is vulnerable to chosen-ciphertext attack. In fact, even if we strengthen the security of cryptographic primitives and require encryption schemes $(\mathcal{K}, E, D)$ and $(\mathcal{K}^0, E^0, D^0)$ to be indistinguishably secure under chosen-ciphertext attack, it still will not help $(\hat{\mathcal{K}}, \hat{E}, \hat{D})$ in attaining chosen-ciphertext attack indistinguishability.

Theorem 6. *Encryption scheme $(\hat{\mathcal{K}}, \hat{E}, \hat{D})$ is not indistinguishably secure under chosen-ciphertext attack.*

Proof. For our scheme, where $\hat{E}_{\hat{k}}(x) = (t, c)$, the distinguishing algorithm proceeds as follows:

1. The algorithm queries the encryption oracle for x and gets ciphertext (t', c').
2. The algorithm queries the decryption oracle for (t', c). This query is allowed and returns some α (note that if the algorithm is input x, then $\alpha = x$).
3. If $\alpha = x$ the algorithm outputs 1; otherwise 0.

Clearly, the advantage of the algorithm is significant. □

However, it is quite easy to modify the scheme so that it becomes indistinguishably secure under chosen-ciphertext attack preserving its homomorphic properties. The underlying idea is to modify the cipher in such a way that it will become infeasible for an adversary with access to a decryption oracle to forge a legitimate ciphertext. One of the possibilities is to augment the ciphertext with a tag containing "Message Authentication Code" (MAC). A ciphertext is considered legitimate if in a pair (c, MAC), MAC is the valid authentication code of c. The simplest way for generating MAC for a ciphertext is to input the ciphertext into a pseudo-random function and use the output as the authentication code.

We define the indistinguishably secure under chosen-ciphertext attack version of encryption scheme $(\hat{\mathcal{K}}, \hat{E}, \hat{D})$ as $(\hat{\mathcal{K}}', \hat{E}', \hat{D}')$ and construct it as follows:

Let $F : \mathcal{K}^F \times \mathcal{Y} \times \mathcal{Y}^0 \mapsto \mathcal{Y} \times \mathcal{Y}^0$ or $F_{k_F}(t, c) = a$, $k_F \in \mathcal{K}^F$, $t \in \mathcal{Y}$, $c \in \mathcal{Y}^0$, $a \in \mathcal{Y} \times \mathcal{Y}^0$ is a pseudo-random function.

Key generation. $\hat{k}' \xleftarrow{R} \hat{\mathcal{K}}'$, where $\hat{\mathcal{K}}' = \hat{\mathcal{K}} \times \mathcal{K}^F = \mathcal{K} \times \mathcal{K}^p \times \mathcal{K}^p \times \mathcal{K}^F$.

Encryption. $\hat{E}'_{\hat{k}'}(x) = (\hat{E}_{\hat{k}}(x), F_{k_F}(\hat{E}_{\hat{k}}(x))) = (t, c, F_{k_F}(c, t)) = (t, c, a)$, where $\hat{k}' = (\hat{k}, k_f) = (k, k_1, k_2, k_F)$.

Decryption. $\hat{D}'_{\hat{k}'}(t, c, a) = \hat{D}_{\hat{k}}(t, c) = D_k(c)$ if $F_{k_F}(t, c) = a$ otherwise the ciphertext is not legitimate and is thus rejected.

According to [Gol04], the encryption scheme $(\hat{\mathcal{K}}', \hat{E}', \hat{D}')$ is indistinguishably secure under chosen-ciphertext attack.

That gives us the following result:

Theorem 7. *Encryption scheme $(\hat{\mathcal{K}}', \hat{E}', \hat{D}')$ is indistinguishably secure under chosen-ciphertext attack.*

Since the only difference between schemes $(\hat{\mathcal{K}}, \hat{E}, \hat{D})$ and $(\hat{\mathcal{K}}', \hat{E}', \hat{D}')$ is simply the authentication tag that is attached to the ciphertext, all the operations feasible under scheme $(\hat{\mathcal{K}}, \hat{E}, \hat{D})$ will remain feasible under scheme $(\hat{\mathcal{K}}', \hat{E}', \hat{D}')$.

Note, that unlike scheme $(\hat{\mathcal{K}}, \hat{E}, \hat{D})$ that does not require search tag for decryption, the scheme $(\hat{\mathcal{K}}', \hat{E}', \hat{D}')$ needs all the members of the triple (t, c, a) in order to check the legitimacy of the ciphertext. That means that if a database is encrypted with scheme $(\hat{\mathcal{K}}', \hat{E}', \hat{D}')$, the complete triples (t, c, a) should be sent to the client, thus tripling the amount of transferred data, compared to the case wherein scheme $(\hat{\mathcal{K}}, \hat{E}, \hat{D})$ is used.

Indistinguishability Up-to-Frequency

As we have already discussed in Section 3.2, when talking about a database privacy homomorphism the indistinguishability alone may not guarantee the security of the encrypted data. In order to perform an operation on the encrypted data, the client often must provide the server with additional input dependent upon the encryption key or the data itself. Recall the example in which a database privacy homomorphism encrypts a table and queries with an indistinguishably secure encryption scheme using two independently generated keys: one for the table and another for the queries. In order to provide the server with the ability to run queries issued by the client, the encrypted table is appended with the key used for encrypting the queries and each query is appended with the key used for encrypting the table. When the client issues a query it encrypts the corresponding SQL statement with the appropriate key and sends it to the server. The server, in turn, by using the key it received with the encrypted table and the key that it has received with the query, decrypts the table and the query, runs the query and sends the result to the client. On the one hand, such database privacy homomorphism indistinguishably encrypts the table and the queries, supporting all possible relational operations. But on the other, this homomorphism gives no security at all as soon as a single operation was performed.

Next we will estimate the amounts of information disclosed when performing operations feasible under the proposed database privacy homomorphism. In our case, all the feasible operations except for exact select and those that are based on it (exact delete, exact update) do not provide the server with any data that depends on the keys or on the encrypted table. As for exact select, we can show that when such queries are processed, nothing except for the frequencies of queried attribute values is revealed to the server. Intuitively, this means that when given an encrypted table and a sequence of queries the server cannot obtain significantly more information about the table than when he is given the encrypted table, knows queried attributes and knows which tuples each query returns. Formally, this notion is captured by Definition 3.2.3.

In Theorems 5, 7 we have already shown that our database privacy homomorphism complies with the first condition of Definition 3.2.3. It remains to be shown that while processing an exact select query over an encrypted table, nothing is revealed but the number of tuples containing the same attribute, which value remains unknown to the adversary.

To be consistent with the notation used in the definition, we introduce algorithm $\hat{E}^*$ that defines the procedure used for mapping exact select queries to the corresponding triples. Also $\hat{E}$, depending on the content, will be defin-

ing either the algorithm that performs the encryption of a single attribute value or the algorithm that performs the encryption of the whole table. And for our database privacy homomorphism $(\hat{\mathcal{K}}, \hat{E}, \hat{E}^*, \hat{D})$ we can formulate the following theorem:

Theorem 8. *For database privacy homomorphism $(\hat{\mathcal{K}}, \hat{E}, \hat{E}^*, \hat{D})$ and for every PPT algorithm A there exists a PPT algorithm A' such that for every table T, every sequence of exact select queries $\bar{\varphi} = (\varphi_1, \ldots, \varphi_q)$, $q = q(n) \leq poly(n)$ and corresponding operations $\bar{\psi} = (\psi_1, \ldots, \psi_q)$, $\psi_i = (\hat{E}_k^*(\varphi_i)$, every polynomially-bounded functions $f(\cdot), h(\cdot)$, every polynomial $p(\cdot)$ and all sufficiently large n,*

$$\begin{aligned}&\Pr\left[A(\bar{\psi}, \hat{E}_k(T), h(T, \bar{\varphi})) = f(T)\right] \\ &\quad < \Pr\left[A'(\bar{R}_{\hat{E}_k(T)}(\bar{\psi}), \hat{E}_k(T), h(T, \bar{\varphi})) = f(T)\right] + \frac{1}{p(n)}\end{aligned}$$

where $\bar{R}_{\hat{E}_k(T)}(\bar{\psi}) = (R_{\hat{E}_k(T)}(\psi_1), \ldots, R_{\hat{E}_k(T)}(\psi_q))$ are sets of references pointing to the tuples of encrypted table $\hat{E}_k(T)$ that constitute resulting sets of exact select queries $\bar{\varphi}$.

Proof. Let $T = \{x_{ij}\}_{i \in \{1,\ldots,m\}, j \in \{1,\ldots,l\}}$ be a table with l attributes and m rows where the values are padded up to the same length and transformed to the binary format (for the brevity of notation we denote attribute a_j (a_j^E) as j (j^E)). Then $\hat{E}_{\hat{k}}(T) = \{\hat{E}_{\hat{k}}(x_{ij})\} = \{(t_{ij}, c_{ij})\}_{i \in \{1,\ldots,m\}, j \in \{1,\ldots,l\}}$ is table T in the encrypted form. Recall, that by $E^*_{k_1,k_2}(\sigma_{j.x})$ we denote triple $(P_{k_2}(x), P_{k_1}(x), j^E)$ that corresponds to the encrypted query $\sigma_{j.x}$. In order to prove the theorem we have to show that for any PPT algorithm A there exists a PPT algorithm A' such that

$$\begin{aligned}&\Pr\left[A((P_{k_2}(x_1), P_{k_1}(x_1), j_1^E), \ldots,\right. \\ &\qquad \left.(P_{k_2}(x_t), P_{k_1}(x_t), j_t^E), \hat{E}_{\hat{k}}(T)) = f(T)\right] < \\ &\Pr\left[A'((R_{\hat{E}_{\hat{k}}(T)}(E^*_{k_1,k_2}(\sigma_{j_1:x_1})), j_1^E), \ldots,\right. \\ &\qquad \left.(R_{\hat{E}_{\hat{k}}(T)}(E^*_{k_1,k_2}(\sigma_{j_t:x_t})), j_t^E), \hat{E}_{\hat{k}}(T)) = f(T)\right] + \frac{1}{p(n)}\end{aligned} \tag{3.9}$$

We build algorithm A' as a composition of algorithms A and B, where B receives $(R_{\hat{E}_{\hat{k}}(T)}(E^*_{k_1,k_2}(\sigma_{j_1:x_1})), j_1^E), \ldots, (R_{\hat{E}_{\hat{k}}(T)}(E^*_{k_1,k_2}(\sigma_{j_t:x_t})), j_t^E), \hat{E}_{\hat{k}}(T)$ as an input, generates a sequence of encrypted queries $(\alpha_1, \beta_1, j_1^E), \ldots, (\alpha_t, \beta_t, j_s^E)$ that is indistinguishable from the sequence $(P_{k_2}(x_1), P_{k_1}(x_1), j_1^E), \ldots, (P_{k_2}(x_s), P_{k_1}(x_s), j_s^E)$ and modifies $\hat{E}_{\hat{k}}(T)$ in such a way, that queries $(P_{k_2}(x_i),$

$P_{k_1}(x_i), j_i^E)$ and $(\alpha_i, \beta_i, j_i^E)$ return tuples that reside at the same positions in the original and modified encrypted tables correspondingly.

As algorithm B begins, it randomly and uniformly chooses from $\mathcal{X}$ bit sequences α_1 and β_1 and for each $\hat{E}_{\hat{k}}(x_{ij_1}), i \in R_{\hat{E}_{\hat{k}}(T)}(E^*_{k_1,k_2}(\sigma_{j_1:x_1}))$ replaces its search tag with a new one computed as $t'_{ij_1} = E_{\alpha_1}(\beta_1)$. Then for each next pair $(R_{\hat{E}_{\hat{k}}(T)}(E^*_{k_1,k_2}(\sigma_{j_p:x_p})), j_p^E), p \in \{2, \ldots, t\}$ algorithm B checks if there exists $s < p$ such that $j_p^E = j_s^E$ and $R_{\hat{E}_{\hat{k}}(T)}(E^*_{k_1,k_2}(\sigma_{j_p:x_p}))$ contains the same references as $R_{\hat{E}_{\hat{k}}(T)}(E^*_{k_1,k_2}(\sigma_{j_s:x_s}))$. If no, then B randomly and uniformly chooses α_p from $\mathcal{X} \setminus \{\alpha_i \,|\, i < p\}$, β_p from $\mathcal{X} \setminus \{\beta_i \,|\, i < p\}$ and for each $\hat{E}_{\hat{k}}(x_{ij_p}), i \in R_{\hat{E}_{\hat{k}}(T)}(E^*_{k_1,k_2}(\sigma_{j_p:x_p}))$ replaces search tag with $t'_{i,j_p} = E_{\alpha_p}(\beta_p)$. If yes, then that means that s-th and p-th queries are identical. Therefore $\alpha_p = \alpha_s$, $\beta_p = \beta_s$ and the corresponding search tags are again replaced with $t'_{i,j_p} = E_{\alpha_p}(\beta_p)$. As the output B returns $(\alpha_1, \beta_1, j_1^E), \ldots, (\alpha_t, \beta_t, j_t^E), \{(t'_{ij}, c_{ij})\}$ that is taken as the input for algorithm A.

Since random values
$A'((R_{\hat{E}_{\hat{k}}(T)}(E^*_{k_1,k_2}(\sigma_{j_1:x_1})), j_1^E), \ldots, (R_{\hat{E}_{\hat{k}}(T)}(E^*_{k_1,k_2}(\sigma_{j_t:x_t})), j_t^E), \{(t_{ij}, c_{ij})\})$ and $A((\alpha_1, \beta_1, j_1^E), \ldots, (\alpha_t, \beta_t, j_t^E)), \{(t'_{ij}, c_{ij})\})$ are identically distributed we can rewrite (3.9) as

$$\begin{aligned} &\Pr\left[A((P_{k_2}(x_1), P_{k_1}(x_1), j_1^E), \ldots,\right. \\ &\qquad \left.(P_{k_2}(x_t), P_{k_1}(x_t), j_t^E), \{(t_{ij}, c_{ij})\}) = f(T)\right] < \\ &\Pr\left[A((\alpha_1, \beta_1, j_1^E), \ldots, (\alpha_t, \beta_t, j_t^E)), \{(t'_{ij}, c_{ij})\}) = f(T)\right] + \frac{1}{p(n)} \end{aligned} \tag{3.10}$$

Suppose now that there exist table T, queries $\sigma_{j_1.x_1}, \ldots, \sigma_{j_t.x_t}$ and function f such that inequality (3.10) does not hold. By then wrapping the expression $A(\ldots) = f(T)$ with algorithm U that outputs 1 when the equality holds and 0 otherwise and performing reductions similar to those that we used in the proof of Theorem 5, it can be shown that there exist a PPT algorithm that could distinguish between pseudo-random permutation P and a random permutation defined on $\mathcal{X}$. The existence of such an algorithm will contradict to the pseudo-randomness of P what concludes the proof of the theorem. □

3.4.5 Indexing and Hashing

Processing of an exact select operation requires to sequentially scan all the tuples of the queried relation. In large databases, this is not efficient, which raises the question of indexing.

If the database is securely encrypted and cannot be decrypted when a query is processed, the usual indexing algorithms are no longer applicable.

For example, it is impossible to perform ordering of the ciphertexts according to their plaintext values. Indeed, let $\prec$ be a binary relation defined on the set of ciphertexts and $E_k(x) \prec E_k(y)$ if and only if $x < y$. Then, given the encryptions of x and y such that $x < y$, the adversary will be able to determine which ciphertext corresponds to which plaintext by simply checking if $E_k(x) \prec E_k(y)$.

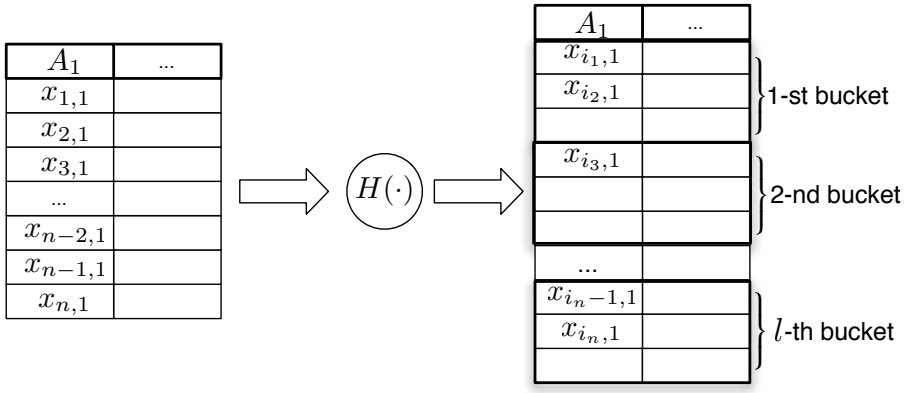


Figure 3.3: Hash-based indexing

The inability to perform ordering of encrypted values of an attribute that should be indexed rules out B+ trees and, in general, all indexing algorithms that rely on the ordering. Hash-based indexing, on the other hand, may remain feasible since a hashing operation is not a subject to such limitations. A basic hash-based indexing algorithm is displayed in Figure 3.3. For building an index of attribute A_1, the tuples of the table are arranged in *buckets* according to the hash values computed as $h_i = H(x_{i1})$ where H is a *hash function* that maps attribute values to integers of the interval $[1; l]$. An attribute value x_{i1} is assigned to j-th bucket if the corresponding hash value is equal to j. Now in order to perform a search for some value x_{i1}, first its hash is computed and then the corresponding bucket is checked, avoiding full scan of the table.

However, hash-based indexing algorithms cannot be directly applied to the encrypted table. The identical plaintext values may be encrypted into different ciphertexts they will have different hashes and therefore will belong to different hash buckets. This problem can be solved by calculating the hash of an attribute value *before* the value is encrypted and sending the ciphertext together with the corresponding hash to the server. Then, when the client issues an exact select query $\sigma_{a_i.x_q}$ with a_i being a hash-indexed attribute, it sends the server not the triple (3.6) but a quadruple (q, k_q, a_i^E, h_q), where

$h_q = H(x_q)$.

Care should be taken when choosing a hash function for indexing encrypted data. If the server knows H, then given ciphertext c and corresponding hash h it can deduce some knowledge about the plaintext. For example, by comparing $H(x)$ with the hash it got from the client it can infer if the ciphertext c is the encryption of x: $H(x) \neq h$ will mean that c is not the encryption of x. However, if the client uses a pseudo-random function for computing hashes, knowledge of the secret key is required for calculating valid hash values. If only the client knows the key, the server is not able to obtain any information about the plaintexts by using their hash values. In addition, if the values of an indexed attribute are highly skewed, such a hash function smoothes out their distribution and uniformly fills the hash-index buckets.

A drawback to such an approach is that it preserves security only if the indexed attributes do not contain any duplicate values. As an example, consider two tables, one with the indexed attribute values being identical for all tuples and another where the values of the corresponding indexed attribute are unique. According to our indexing algorithm, the server partitions attribute values into buckets based on hashes precomputed by the client. Since the same values will produce the same hashes, the indexed attribute of the first table will produce a single bucket and the indexed attribute of the second table with high probability will produce more than one. Using this information the adversary will be able to distinguish these two tables even if they are encrypted with a secure encryption scheme. According to Definition 2.5.3 the resulting scheme is not indistinguishably secure. An analogous analysis is applied to the more common case when the indexed column contains only some duplicates. Therefore, only key attributes can be securely indexed this way without having the index leak anything about the distribution of plaintext attribute values.

3.4.6 Comments on the Scheme Described by Yang et al.

In [YZW06] Yang et al. propose their own security model for privacy-preserving query protocols. First, they introduce the notion of the *minimum information revelation* of exact select query q issued to table T, which is the set of coordinates of the cells satisfying the condition of the query. Denoting the minimum information revelation by $R_T(q)$, they present their version of the definition of the query protocol revealing nothing but frequencies:

Definition 3.4.1. *A one-round query protocol reveals nothing beyond the minimum information revelation if for any polynomial* poly() *and all sufficiently large n, there exists a PPT algorithm S (called a simulator) such that for any* $t < \text{poly}(k)$*, any polynomial-size circuit family* $\{A_n\}$*, any polynomial* $p(\cdot)$*, and any sequence of exact select queries* $\varphi_1, \ldots, \varphi_q$*,* $q = q(n) \leq poly(n)$

$$|\Pr[A_n(\bar{E}_k^*(\bar{\varphi}), E_k(T)) = 1] - \Pr[A_n(S(R_{E_k(T)}(\bar{E}_k^*(\bar{\varphi})), E_k(T)), E_k(T)) = 1]| < \frac{1}{p(n)}$$

However, this definition contains one serious flaw: it does not impose any requirements on the security of the encryption scheme that is used to encrypt table T. As an example, consider a protocol that performs no encryption at all and operates with plaintext tables and queries. In such protocol for any table T (query φ_i) $E_k(T) = T$ ($E_k^*(\varphi_i) = \varphi_i$) it is trivially to build simulator S that by observing $E_k(T)$ and $R_{E_k(T)}(\varphi_1^*), \ldots, R_{E_k(T)}(\varphi^*)$ reconstructs queries $\varphi_1, \ldots, \varphi_t$ and returns them with $E_k(T)$ as the output. Clearly, with such a simulator, the difference of the probabilities from Definition 3.4.1 will be 0.

The encryption scheme and the querying algorithm proposed by Yang et al. exploits the approach similar to the one we proposed in Section 3.4.2. But by proving that the described query protocol satisfies Definition 3.4.1, the authors state that the protocol reveals only a number of tuples sharing the queried value and the queried attribute. As we have shown above, this definition, actually, says nothing about the strength of the encryption and the level of security provided by the protocol.

Also Yang et al. mistakenly assume that their protocol returns those, and only those tuples that satisfy an issued exact select query. By applying the same reasoning as we did in Section 3.4.3, it can easily be seen that the protocol may allow erroneous tuples to be included in the resulting set.

3.5 Summary

If a database production system is deployed, no one questions the virtue of a sound theory of databases on which the system is based. If an insecure network connection is used between client and server for sensitive applications, reliable encryption and authentication mechanisms are used to protect the user against attacks. However, solutions for the problem that the database server itself goes adversarial have so far often been based on much lower standards: rather than focusing on acceptable worst-case bounds of security, researchers have been more concerned with minimizing their performance overhead.

We have given several new security definitions for database privacy homomorphisms and proposed a cryptographic technique that allows for secure processing of encrypted data in the presence of a non-trusted database service provider. We have shown that a combination of trust and appropriate technical means can protect a client from undesired security breaches if the range of operations outsourced to the service provider is limited.

In particular, we presented a database privacy homomorphism that is secure in a relaxed but still rigorous and plausible sense under widely accepted cryptographic assumptions, together with a proof of security.

We have also designed an encryption scheme that allows the secure outsourcing of a substantial subset of relational database operators: exact select, Cartesian product, projection, exact update, exact insert and exact delete. Our approach represents the first solution to the database outsourcing problem that is provably secure and supports such an extensive set of relational operators. We have conclusively proved the security of our scheme and showed how to reduce the probability of having erroneous tuples in the answer to an exact select query to a negligible level. Moreover, we have presented some thoughts on how to securely perform indexing in the context of encrypted database outsourcing.

We hope that our approach of first establishing suitable security definitions first and then finding encryption schemes that satisfy them will prove valuable for future research in this area, in particular in light of our constructive results.

Chapter 4

Access Control to Outsourced Databases

In the previous chapter we described how to outsource database to a non-trusted ASP without compromising privacy of data that are stored there. Assuming that the security of the outsourced database can be ensured, the next logical step is to provide a means for managing discretionary access to these data. In this chapter, we describe a solution that enables users to define access rules to the data they submit to the securely outsourced database. We compare the proposed solution with existing analogues and analyze its practicability by evaluating results of conducted performance experiments against a real life scenario where the solution could be applied.

4.1 Introduction

In this section we describe the problem of providing discretionary access control to a securely outsourced database, review related work and briefly discuss our main results related to this problem.

4.1.1 Motivation and Problem Statement

Consider a 2-party database outsourcing scenario in which multiple clients collaboratively provide and query some sensitive data in a database that is outsourced to an ASP – Figure 4.1. If the clients do not trust each other, they might want to control the access to the parts of the data they own and be able to selectively assign certain read and write access permissions to other clients. Usually the problem of database access control is formulated as the ability of a database administrator to manage access privileges of database users

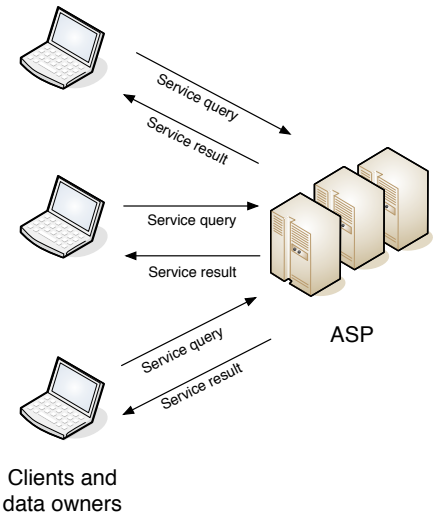


Figure 4.1: Multiple clients accessing outsourced database

[RG02]. In most of the *de facto* industry standard database management systems the discretionary access control is implemented by means of privileges, roles, views, stored procedures and virtual private databases [BS05]. Additionally, there exist a number of not so widely spread techniques such as query rewriting [SW74] and multilevel security databases [BL76]. In all of these approaches, it is assumed that the database administrator has full access to the data.[1] However, when the database is outsourced, the data owners should be aware that their data will be stored on servers that are owned and maintained by the party over which they have no direct control. In the case when both the ASP and the other clients are not trusted, the following problems arise: (*i*) the ASP can unnoticeably grant read and write permissions to clients not having appropriate access rights; and (*ii*) if the database stores sensitive data and is securely outsourced, for example, employing techniques described in the previous chapter, the inability to distinguish between tuples of a database table makes it impossible for the ASP to determine whether a client has read or write access to a given tuple.

As a motivating example we consider an outsourcing of some product-related data that is owned and shared by different parties constituting a product supply chain. Assume a supply chain in which each product is assigned a unique id number. Such item level identification allows the supply chain participants to associate with each item a detailed description that, if

[1]The administrator may have this access indirectly by having full control over the database management software and the hardware on which the software is installed.

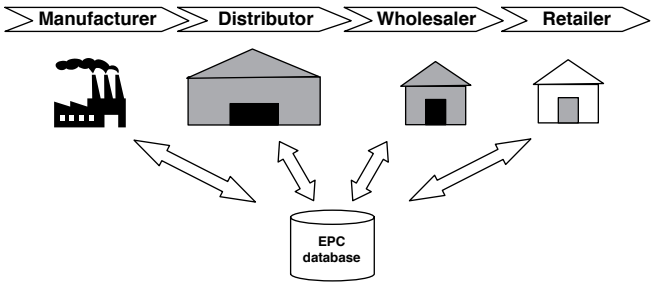


Figure 4.2: Supply chain with shared centralized EPC database

stored in a centralized remote datastore, can be efficiently accessed and used by all the supply chain participants – Figure 4.2. Bearing in mind a scenario in which the supply chain is RFID-enabled and the id numbers are stored in RFID tags, we will refer to such id numbers as to Electronic Product Code (or EPC) identifiers[2] [EPC06].

It very likely that often the supply chain participants will want to selectively restrict access to the product data they submit to the database: certain portions of the product data may be considered sensitive and their owners might want to prevent some of the partners from seeing them. As an example, consider a manufacturer who needs to keep track of a factory that produced a product and stores the id of the factory as a part of product data. However, in order to prevent discrimination of the products based on their origin, the manufacturer considers this information private. Furthermore, suppose the manufacturer agrees to share this information with the wholesaler, but the retailer and the logistic companies that also provide services to competitors should have no access to it.

4.1.2 Related Work

There are a number of existing solutions suggesting methods for implementing discretionary access control to a database that is outsourced to a non-trusted ASP. Most of them deal with access control in a tree-like hierarchy [AT83; HL90; HY03; MTMA85; Tan06] that is a special case of a general access control problem: a client has access to all objects that are owned by clients located on lower hierarchy levels and has no access to objects owned by clients located on higher hierarchy levels. All these solutions are based on

[2]RFID (Radio Frequency Identification) is a technology that provides an efficient way for object identification by storing an EPC identifier in an RFID tag that can be remotely queried by an RFID reader.

the same idea that is to selectively encrypt table tuples with keys that are derivable from each other according to the hierarchy: keys corresponding to the lower hierarchy levels can be derived from keys assigned to a higher level, but doing the same in the opposite direction is computationally infeasible.

However, very often access permissions of the clients cannot be arranged as a hierarchy. In the supply chain example, some logistic companies may be selectively provided with a detailed information about items they transport, some retailers may be allowed to access a production specific information and others not. In [DdVF+05] Damiani et al. consider a very similar problem and present a solution that allows for implementing access control in a multi-user database supporting arbitrarily defined tuple-wise access permissions for the users. Using an example, we briefly outline their approach.

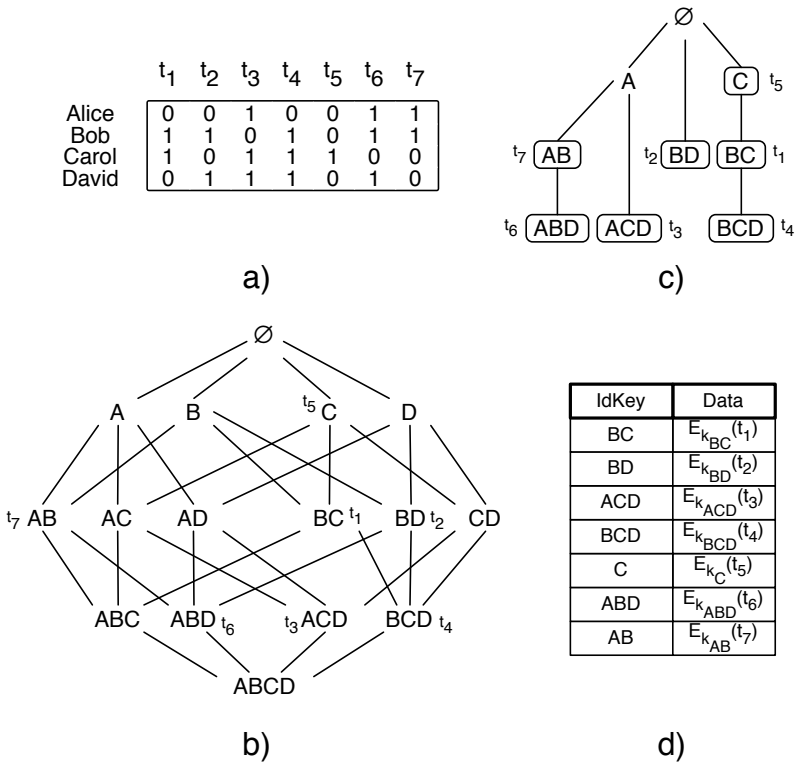


Figure 4.3: Access matrix (a), corresponding hierarchy graph (b), its tree representation (c), and encrypted tuples (d)

The proposed solution considers a table in a multi-user environment where each user is assigned a set of privileges that give her access to a subset of tuples. In our example we consider a table consisting of seven tuples $t_1, \ldots, t_7$ with permissions defined by an access matrix (Figure 4.3a) where

ijth element is equal to 1 if ith user can access tuple t_j and to 0 otherwise. The set of access privileges is then represented as a directed acyclic graph (Figure 4.3b), which by means of a heuristic algorithm is transformed into a hierarchy tree (Figure 4.3c). Each node of the tree is assigned a key that is produced as an output of a pseudo-random function that uses a name of the parent node as its key and takes an argument that is a key assigned to the parent node: $k_{n_i} = f_{\text{name}(n_i)}(k_{n_j})$ where node n_j is a parent of node n_i (e.g. $k_{\text{AB}} = f_{\text{AB}}(k_{\text{A}})$). For the root node, the key is randomly and uniformly generated. The keys are used to encrypt the corresponding tuples (Figure 4.3d). By making the keys derivable from each other, the algorithm reduces the number of keys a user must store in order to be able to decrypt tuples to which she has access rights.

However, there are a number of issues impacting the security and performance of this solution. As the authors themselves note, changes in access rights, users and objects often lead to changes in the structure of the hierarchy tree resulting in the necessity to re-encrypt the data and redistribute the keys. If such changes happen relatively often, the performance of the system may be seriously affected. As for the security, appending each tuple with the lists of users that have access right on it exposes the relationships between the users. Finally, the existence of the root key implies the existence of a party that has full access to the data, limiting the applicability of such an approach.

Thus there is still a need for a solution that overcomes the aforementioned problems and allows for efficiently and flexibly managing access rights to the outsourced database.

4.1.3 Our Contribution

In this chapter, we address the problem of discretionary access control for a database that may contain sensitive data and is outsourced to a non-trusted ASP. We present a practical solution that allows the data owners to selectively restrict read access to the outsourced data and protect it from modification by parties that have no write access permissions to it. Compared to the existing solutions, our approach does not reveal the relationships between the parties and allows efficient dynamic assignment of read permissions. Moreover, compared to most existing solutions that are applicable only when permissions of the clients constitute a hierarchy, our solution supports arbitrary layouts of the access permissions.

We perform an experimental evaluation of our approach and discuss its practical applicability on the example of a database that stores product information about objects of an existing supply chain consisting of several

thousand parties.

4.2 Read-Write Access Control

In this section, we describe how to enable users of a securely outsourced database to define read-write access rules to the data they submit to the database. We also discuss practical applicability of the proposed solution by analyzing results of performance experiments on the example of the supply chain of an existing company.

4.2.1 Basic Idea

In the previous section, we have listed a number of approaches that solve the problem of discretionary access control to the outsourced data. The primary aim of these studies is to allow the data owners to maintain exclusive control over the access to their data using cryptographic methods while minimizing the amounts of keys they have to manage. In many respects it is done at the expense of narrowing the original problem to a more specific case, in which permissions of the clients allow for arranging them in a hierarchy, the relationships between the clients are open information and the data and the access permissions are static. While seeming reasonable in some scenarios, in many cases it might be necessary to handle arbitrary patterns of clients' relationships, keep these relationships confidential and quickly react to structural changes. We base our solution on similar principles by also proposing to encrypt database tuples with different keys that are distributed between the users in such a way that they can only access data to which they are granted appropriate permissions. However, our primary objective is not to reduce the number of the keys but to be able to handle arbitrary dynamic patterns of the access permissions, to not reveal information about "User A grants access to this tuple to User B" relationships and to make the solution efficient and easy to implement.

We assume that the data is stored in a single table and that there is at least one attribute whose values are not private, and uniquely identify the tuple in the table (e.g., EPC identifier in the supply chain example). The values of this attribute may stay accessible to all clients and therefore may be be used in `WHERE` part of exact select queries issued by the clients.

We start with a simplified version of the algorithm that might be applicable to a "linear" configuration of the supply chain and conclude with a version that can be applied for general configurations.

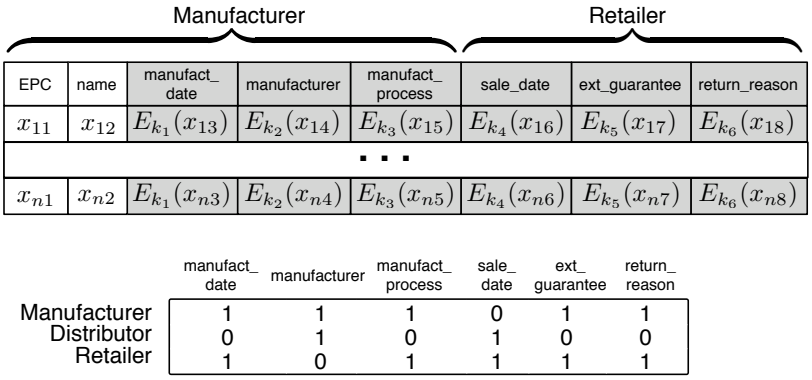


Figure 4.4: Table with product data and its access permissions (simplified scenario)

4.2.2 Read Access Control in a Linear Scenario

The linear scenario assumes that each attribute of the table has a single owner, but can be accessed by any of the clients. Applied to the supply chain example, such a layout can arise when the supply chain has a linear structure: the manufacturer, the distributor, the retailer, etc. – each of these stages consists of a single participant (e.g., the supply chain as depicted in Figure 4.2). Consider a table that consists of a primary key attribute $attr_{\text{pk}}$ and a set of attributes $A = \{attr_1, \ldots, attr_n\}$. The primary key can store EPC identifiers and the rest of the attributes can be filled with the product-related data. Let $U_1, \ldots, U_n$ be the clients accessing the outsourced database. Every client U_i owns a subset of attributes $\mathcal{A}_i = \{attr_{i_1}, \ldots, attr_{i_{n_i}}\}$, $\mathcal{A}_i \subset A$ and is responsible for providing values for these attributes (e.g., the manufacturer may own attributes `EPC`, `factory`, `manufacturing_date`, etc. and fill them with appropriate values for each produced item that is tagged with an RFID chip). Due to the linear structure, there are no attributes that are shared between the participants: $\mathcal{A}_i \cap \mathcal{A}_j = \emptyset$.

In order to allow the participants to control access to the owned data in such linear case, it is sufficient to introduce an attribute-wise access control mechanism. It can be easily implemented through an attribute-wise encryption of every tuple, where each attribute value is encrypted with a key, generated by an attribute owner: user U_i owns attribute $attr_{i_k}$ and encrypts its values with key k_{i_k} using a symmetric encryption algorithm. If user U_i decides to grant user U_j access to the values of attribute $attr_{i_k}$, user U_i shares key k_{i_k} with user U_j.

Figure 4.4 illustrates a possible table storing product-related data for a

linear supply chain that includes a manufacturer, a distributor and a retailer. The table has seven attributes: five are owned by the manufacturer, three are owned by the retailer, and the distributor does not own any of the attributes and can only access data entered by the manufacturer and the retailer. The attributes containing the EPC identifiers and names of the objects are freely accessible. Access to the six shaded attributes should be restricted and therefore the owners generate encryption keys (one per owned attribute) and use them for encrypting the corresponding attribute values: the manufacturer generates keys k_1, k_2, k_3, the retailer generates keys k_4, k_5, k_6. According to the access matrix these keys are distributed in the following manner: the manufacturer receives keys k_5, k_6, the distributor receives keys k_2, k_4, the retailer receives keys k_1, k_3. After the keys are distributed, the users can query the table for the tuples containing the product data and that are identified by the corresponding EPC identifiers. Then, using the owned or the obtained keys, they can decrypt the attribute values to which they have read access.

4.2.3 Read Access Control in a General Scenario

Now we extend the described algorithm so that it may provide discretionary access control for general scenarios where clients can own arbitrary attribute values and provide access to these values to any other client. In the supply chain example this means that the supply chain participants constitute an arbitrary weakly connected graph. An example of such a supply chain is illustrated in Figure 4.5.

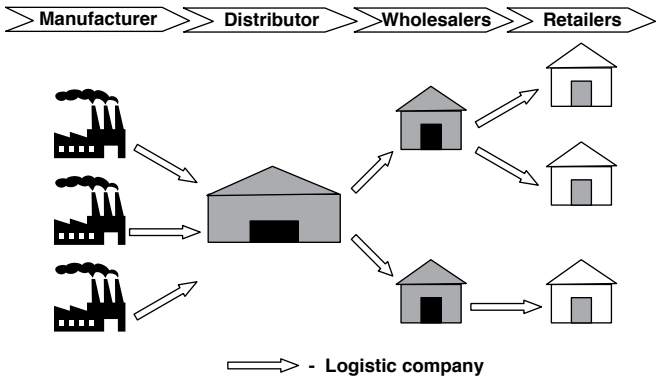


Figure 4.5: Model of a supply chain

Unlike the scenario where the supply chain has a linear structure, the attribute-wise access control is not sufficient here since each stage (manufacturing, distributing etc.) can include more than one participant. That

means that values of an attribute can be submitted by different parties that may be setting different access permissions on them. Moreover, they may want to restrict each other from accessing the values they submit. This rules out the possibility of assigning ownership over the attributes to groups of the participants and applying the solution described in the previous section. In order to implement the discretionary access control in such a general case in addition to the attribute-wise access control, the tuple-wise access control should be provided as well. In other words, the clients should be able to define cell-wise access rules.

A straightforward way to implement the cell-wise access control is to have the attribute values encrypted by the owners, distributing the keys according to the access permissions. However, now the values of the same attribute can have different owners and, if they are encrypted by a secure encryption scheme, the ciphertext cannot help in determining the owner of an attribute value. Therefore, to let the clients know which key should be used for decryption, each attribute value should be accompanied by an id of the client owning the value.

Such an approach may require users to store large numbers of keys, but considering a relatively short bit length of the keys used by symmetric encryption schemes[3], it should not be considered a significant drawback.

Albeit quite simple and efficient, such an approach fails to securely protect attribute values from parties that do not have appropriate read permissions. Sometimes the identifier of the owner, which accompanies every encrypted value, can be used for guessing the encrypted values. Even if the identifier does not directly refer to the owner, it still allows for singling out the values submitted by the same client. Such information can be used for launching a statistical attack on the encrypted attribute values. As an example of such an attack, consider a supply chain that includes one major and several minor manufacturers. By picking a set of randomly selected tuples and counting the frequencies for the tuples that have the same owner for an attribute that contains manufacturer's data, an adversary can assume that the values with the highest frequency were submitted by the largest manufacturer. If the adversary has knowledge about the shares of other manufacturers, she can also make similar assumptions for the rest of the tuples.

To make the algorithm sustainable to such attacks, the owners of the encrypted values should not be exposed. However, in such a case, the clients will be not able to single out the valid key for an encrypted value and, therefore, will have to perform a brute force search among the keys that could

[3]According to NIST recommendations AES encryption algorithm with 128 bits key is sufficient for encrypting data with security lifetime beyond the year 2030 [BBB+07].

be used to encrypt this value. We will call such keys *candidate keys*. The search is performed by sequentially trying to decrypt the encrypted value with each candidate key until one of the keys produces a valid plaintext. To make the plaintext efficiently recognizable and to speed up the search, an attribute value x can be prepended by the value of the corresponding primary key pk that is separately encrypted with the same key $(E_k(pk)|E_k(x))$ or, in case a block cipher in one of the chaining modes is used, padded up to be a multiple of the cipher's block size and encrypted together with x $(E_k(pk_{padded}|x))$. Now to find a valid key it is sufficient to be decrypting only a part of the ciphertext containing the value of the primary key. Herein we will be assuming the use of the block cipher in one of the chaining modes and consider the approach where the primary key value is padded up and concatenated with the attribute value.

To summarize, consider clients $U_{i_1}, \ldots, U_{i_l}$ that are a subset of all clients that can submit values of attribute $attr_j$ and $k_{i_1 j}, \ldots, k_{i_l j}$ are the keys with which these users encrypt the values they submit: key k_{ij} is generated by user U_i and attribute value x submitted by U_i is stored as $c = E_{k_{ij}}(pk_{padded}|x)$. Suppose client U has read access to the values of attribute $attr_j$ submitted by $U_{i_1}, \ldots, U_{i_l}$ and, therefore, knows the corresponding keys. When user U receives tuple $\langle \text{PK} : pk, attr_1 : c_1, \ldots, attr_n : c_n \rangle$ from the ASP, in order to read the value of attribute $attr_j$ she tries to decrypt c_j by sequentially applying the decryption procedure with keys $k_{i_1 j}, \ldots, k_{i_l j}$ to the part of the encrypted attribute value that contains the padded value of the primary key. The client stops either when there is a key that produces the valid pk or after the last key was tried. The former means that U has read access to this value and the found key can be used to decrypt the rest of the ciphertext, the latter means that the access was denied.

The necessity for looking through the set of candidate keys inevitably increases the time needed to process a tuple. We postpone a detailed discussion on the performance and applicability of our approach to the next section and only mention that it is comparable to the performance of the algorithm from [DdVF+05], which we described in Section 4.1.2. In the worst case, this algorithm requires *Number of Clients* computations of a pseudo-random function plus decryption of the encrypted attribute value, whereas the worst case scenario for our approach supposes *Number of Clients* decryptions of the primary key value plus a decryption of the encrypted attribute value.

One issue remains to be dealt with: the ASP can still identify the owners of the submitted values by being able to track the origin of an insert or update query. To prevent this the clients may deploy an onion routing network [DMS04] (also known as Tor), which will make the communication between them and the ASP anonymous. Such a network is relatively easy to

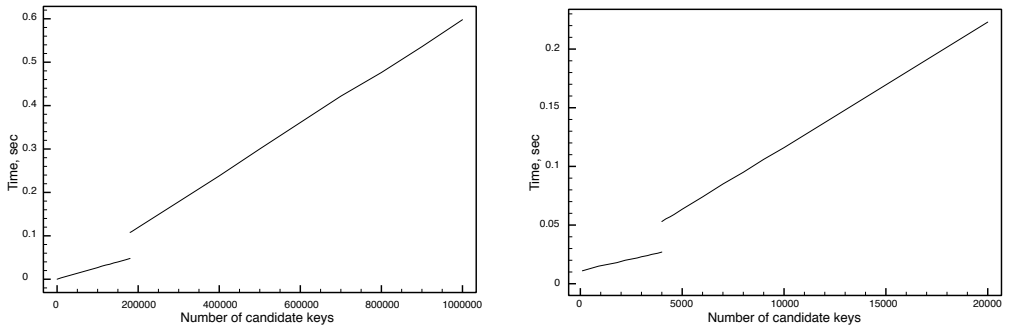


Figure 4.6: Time elapsed for N decryption on a PC (a) and a handheld RFID reader (b)

configure and maintain, and the only negative side effect is minor communication latencies. However, since the anonymity is required only for inserts or updates, requests for data stored in the database, which will constitute most of the traffic, can still be efficiently carried out using conventional routing protocols.

4.2.4 Performance Tests

The average time a client needs to process a tuple with read access permissions defined as described in the previous section is proportional to the joint number of the owned candidate keys for all protected attribute values she wants to read. To estimate possible delays and to see how they may affect the performance of the system we simulated the permission verification process for an attribute value and measured the average access time. The tests were performed on a PC and on a handheld RFID reader that has sufficient computational power to work with cryptographic primitives. The PC had Pentium IV 2.80 GHz CPU with 1024 MB of RAM and was running Windows XP; as a handheld RFID reader we used Nordic ID PL3000 supplied with Sharp 200 MHz ARM processor and 32 MB RAM running Windows Mobile 5.0. Such an environment allowed us to examine an applicability of our approach to the supply chain example in which a tuple is accessed once an RFID reader of some client reads the corresponding EPC identifier from the RFID tag. A PC can be used for controlling one or more stationary RFID readers, while a handheld reader can process the data autonomously.

Each test run comprised N decryptions of a 128-bit ciphertext block. Each decryption was performed with one of the candidate keys and the result of each decryption was compared with the valid plaintext. As the encryption

algorithm, we used AES cipher in CTR mode with 128-bit key. The test application was written in C and used LibTomCrypt cryptographic library for AES implementation. The results of the tests are presented in Figure 4.6.

The charts display time (Y-axis) required to check a certain number of candidate keys (X-axis). The change of the slope on both charts is caused by different modes of operation of the search procedure. The gentle slope corresponds to the mode where at first N cipher variables were initialized with candidate keys and then sequentially used in a cycle where the blocks of ciphertext containing the value of the primary key were decrypted and the results verified against the valid primary key value. The steep slope corresponds to the mode where a single cipher variable was reinitialized with a new key at each iteration of the cycle. Since the preinitialization of the cipher variables has to be performed only once, in the former mode the search is performed more than twice as fast as in the latter mode. To avoid I/O delays the preinitialized cipher variables have to fit into RAM of the device, thus making it necessary to switch to the second mode when N was becoming too large: the PC with 780 MB of RAM and the RFID-reader with 19 MB of RAM available for the test application could fit at most 180000 and 5000 preinitialized cipher variables in the RAM.

Let's interpret these results for a hypothetical supply chain that consists of S stages (e.g., production, distribution, wholesaling, retailing, etc.) and has N_i participants operating at each ith stage. A participant operating at ith stage submits values to a number of attributes, M_i of which are assigned some access permissions. For example, the table displayed at Figure 4.7 may correspond to a supply chain with $S = 2$, $N_1 = 3$, $N_2 = 3$, $M_1 = 2, M_3 = 3$ (the protected attributes are grayed). In order to check access permissions for an attribute value submitted at ith stage at most N_i candidate keys have to be tried. To check access permissions for attribute values that are defined at the same stage (e.g., `manufacturer`, `manufact_date`) it is sufficient to determine a key only for one of them since the rest has the same owner. Thus, if there is a supply chain participant with full read access to the product data stored in the extracted tuple (the worst case scenario), in order to read values of all protected attributes, she will have to try at most $N_1 + \ldots + N_S$ candidate keys or $(N_1 + \ldots + N_S)/2$ keys on average.

To put some real numbers behind this hypothetical scenario we consider a supply chain of GERRY WEBER International AG – a large European apparel producer. At the time of writing, their supply chain had the following structure (the information was obtained in a private communication):

- producers: about 180
- wholesalers: 1 (GERRY WEBER International AG)

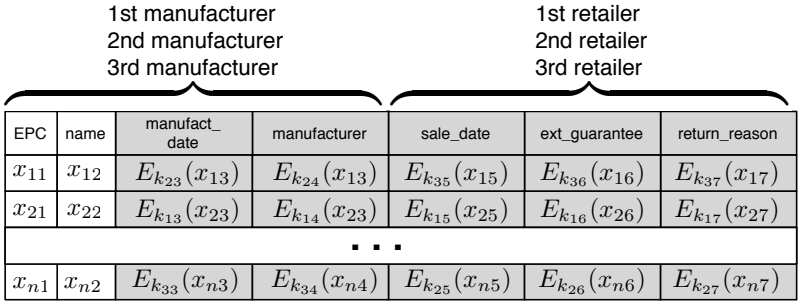


Figure 4.7: Product data

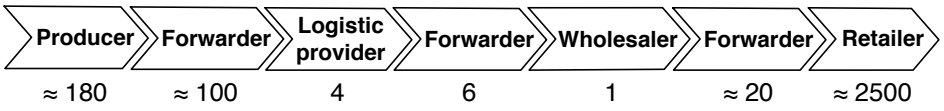


Figure 4.8: Structure and numbers of participants in the supply chain of GERRY WEBER International AG

- retailers: about 2500
- forwarders:
  - from producers to consolidation points: about 100
  - from consolidation points to distribution center: 6
  - from distribution center to stores: 20
- logistics service providers: 4

The supply chain structure is depicted in Figure 4.8.

Thus, $S = 7$, $N_1 + \ldots + N_7 \approx 2811$. Therefore, to fully read a tuple with product data a supply chain participant with full read access in the worst case will have to try about 2811 candidate keys and about 1406 candidate keys on average. According to the results of the performance tests, when using analogous equipment, on average it will take about 0.4 and 10 milliseconds on the PC and the handheld RFID reader correspondingly. Thus, when all the computations are performed on the handheld reader, accessing the product data for 100 items will take about one second and during the same time interval the PC can process about 2500 items, allowing for it to be used as a controller for several stationary RFID readers.

4.2.5 Write Access Control

So far we have only described the technique that allows for controlling read access permissions. However, it is often also necessary to be able to protect the outsourced data from unauthorized changes. For an outsourced database, if no countermeasures are implemented, the database clients or the ASP can modify the data stored in the table and the data owner has no way of preventing or being informed about such modifications.

Since the clients have no physical access to the database, it should be at least possible to detect such changes. A standard technique for verifying integrity of a value is to digitally sign it [RSA78]. The same can be done with the attribute values. Let (G, E', D') be an asymmetric encryption scheme where G, E' and D' are key generating, encryption and decryption algorithms. Algorithm G generates a pair of keys (s, v) where s is a private key used for signing and v is a public key used for signature verification. To protect value x from the unauthorized modification, the owner of the value computes its hash value $d = h(x)$ where h is a cryptographically strong hash function, and produces digital signature sig by encrypting the hash value with encryption algorithm E' and private key s. The signature is appended to the value: $(x|sig) = (x|E'_s(h(x)))$. Now every client given a signed attribute value $x|sig$ is able to check whether this value is authentic. To do so, a client computes the hash $h(x)$, decrypts the signature with public key v and compares the result of the decryption with the computed hash value: $h(x) \stackrel{?}{=} D'_v(sig)$. The cryptographic strength of the hash function implies its collision resistance, thus making it infeasible for a malicious user to pick a value that would produce the same hash. The fact that key s is known only to the owner of the value ensures that only the owner could compute the signature. Thus, the authenticity of the value is confirmed when the equality holds true. Otherwise the value was forged.

In addition to the ability to ensure the integrity of the value, it is also important to fix the position of the value in the table. If the digital signature is based only on the value, a malicious client or the ASP may unnoticeably interchange values of tuples or attributes. For example, it may be possible to substitute the manufacturing date of a product with expired shelf life with the manufacturing date of a product that is still usable. To prevent such modifications, the hash value should be computed from a combination of the value, its attribute name and corresponding primary key: $h(pk|attr_j|x)$.

Thus, a value of attribute $attr_j$ that is owned by client U_i and is protected from unauthorized read/write has to be stored in the following form: $(E_{k_{ij}}(pk_{padded}|x)|E'_{s_i}(h(pk|attr_j|x)))$. Key v_i is made publicly available and key k_{ij} is distributed to the clients with read permissions for this value. Af-

ter a client with appropriate read permissions determines that the value is encrypted with key k_{ij} she can use v_i to verify the signature for ensuring its authenticity.

4.3 Summary

We have described a technique that allows clients to define read/write access permissions to an outsourced database. Our technique allows for dynamically granting read and write access permissions to the existing or new users without the necessity to re-encrypt the data, does not impose any communicative overhead, is easy to implement and is lightweight enough to be supported by handheld computer devices.

We have also conducted performance tests that have shown efficiency and practicability of the proposed technique. The practicability has been confirmed by analyzing the results of the performance tests against the scenario in which we have considered an outsourced database storing data about products of a large existing supply chain. The considered supply chain consists of several thousands participants where each participant is granted access to the outsourced database and is given a set of access permissions for the data stored there. We believe that a growing popularity of RFID-enabled supply chains that assume sharing of product data using architectures similar to the one we have considered in this chapter makes our results relevant for practical applications.

Chapter 5

Secure Outsourcing of Content-Based Routing Operations

In this chapter, we examine a different setup that nevertheless is related to the problem of secure database outsourcing. Assuming a non-trusted provider that enables content-based routing, we examine the possibility of performing such a routing while preserving the confidentiality of routed messages. We introduce a model that formalizes this requirement and captures the generally accepted concept of such an architecture. We then examine the possibility of constructing a content-based routing system that can provide both confidentiality and efficiency within such a model. Upon discovering that these goals cannot be achieved together, we propose a slight modification of the model and illustrate that with such changes confidentiality and efficiency become feasible.

5.1 Introduction

In this section, we describe the problem of secure content-based routing, review related work and briefly list our main results related to this problem.

5.1.1 Motivation and Problem Statement

Consider a 3-party outsourcing architecture comprising of clients, data owners and service providers. The clients are interested in some content that is provided by content-providers and delivered to the interested clients by a service provider that plays a role as a routing mediator.

As a motivating example, consider a system in which consumers register to a notification service with a standing query for events of interest. Event producers push event messages to the notification service that forwards them to consumers with corresponding queries. This communication scheme implements the observer design pattern and can be applied in applications that need to monitor a state of various systems [FMG02]. Furthermore, event-based systems leverage flexibility due to the indirect addressing scheme of event-based communication. These properties make event-based systems appealing in a range of application domains, such as enterprise application integration or ubiquitous computing [CBB03; RM04]. An event can be defined as "Any happening of interest that can be observed from within a computer" [MFP06]. This may be, for instance, data captured by sensors or a transaction in an ERP system. Applications use such event sources to discover situations of interest, react to exceptions, or trigger actions of a workflow.

Content-based routing is a promising paradigm for implementing such architectures. It provides an efficient and scalable way for delivering published events to multiple parties. A typical content-based routing system consists of a number of *publishers* that generate and publish content as a sequence of events, a number of *subscribers* that can subscribe for certain types of events, and a network of *brokers* that are organized in a peer-to-peer manner and are responsible for routing the published events to the interested subscribers. An example of such a network is displayed in Figure 5.1.

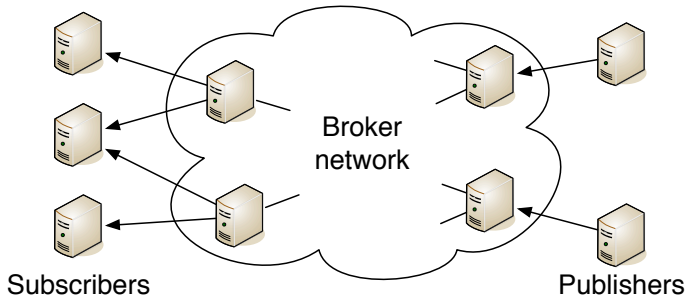


Figure 5.1: Content-based routing network

As it can be seen, the architecture of the content-based routing system includes multiple parties, thus making security an important concern. Requirements to ensure service availability, authenticity of the parties, authenticity and integrity of the routed events can be satisfied by the existing network security solutions (e.g., by relying on TSL protocol for network communications). Yet there exists no such straightforward way to ensure the confidentiality of the routed events. When compared to conventional routing

protocols where senders explicitly indicate addresses of recipients and routing nodes simply look for paths leading to them, in content-based routing systems the brokers examine the content of the event message and construct the paths depending on which subscribers are interested in that content. That again brings the problem of secure service outsourcing. On the one hand, in order to be able to provide the service the broker network should have the ability to undertake actions that are dependent on the content of the event messages. But on the other hand, there can be a requirement to keep the content of the routed messages confidential, if the brokers are not trusted.

This brings us to the problem that is similar to the problem of secure database outsourcing: usually, in order to perform a routing, a broker must compare an arrived event message against a set of filters that allow him to determine the next destination point of the message. Such an operation can be viewed as a processing of SELECT queries represented by the filters over the data represented by the routed messages. Thus, the ability to securely outsource and query databases may be considered an approach for constructing a content-based routing system preserving events confidentiality against the network of not-trusted brokers.

However, before building a routing protocol that can provide events confidentiality, one should formally define its functional and security requirements. The definitions should, on the one hand, capture generally accepted architecture and capabilities of content-based routing systems and, on the other hand, fit into the provable security paradigm.

5.1.2 Related Work

When developing a content-based routing system, a number of security considerations should be taken into account. In [WCEW02], Wang et al. define security requirements for such a system and describe possible security threats. Among them are authentication of the system's members, integrity of the transmitted events, service integrity, event confidentiality and users' anonymity.

While most of the mentioned problems are not new and can be solved using standard network security mechanisms (digital signatures and certificates, onion routing, etc.), as we have noted earlier, the problem of preserving the confidentiality of the events has obvious implications.

In [OP01], Opyrchal et al. consider a content-based system which subscribers are not trusted and propose a solution to the problem that is similar to the one we have examined in the previous chapter: how to perform content-based routing ensuring that published events will be read only by subscribers

with appropriate permissions, given that each subscriber is authorized to access messages corresponding only to certain types of events. To solve this problem, they propose encrypting the event messages at the end-points of the network using different keys and distribute the keys between the subscribers in such a way that the messages could only be decrypted by the subscribers with appropriate access permissions. Additionally, the authors propose several caching techniques that allow for reducing the number of stored keys and decryption operations.

An important assumption the authors make about their settings is that the broker network is trusted. Since this is not always the case, there is a number of works that consider an orthogonal problem in which it is required to preserve confidentiality of the routed event messages against non-trusted brokers.

A framework named EventGuard that allows for withstanding DoS attacks, ensuring integrity and confidentiality of the events and authenticity of the publishers and the subscribers is described in [SL05]. By being able to reach goals related to integrity and authenticity through relying on digital signatures, the confidentiality of events that is provided by the framework towards non-trusted brokers is very limited. According to the framework, the content of the event is deterministically encrypted producing a so-called token. Such approach allows for implementing event routing with equality-based subscriptions. However, it guarantees confidentiality only if a broker receives and routes at most one token. In case the broker observes multiple tokens, the system becomes vulnerable to the statistical attack since the same event values will be substituted by the same tokens.

A similar approach is proposed in [LLS04]. But unlike the previous solution, it does not only allow for equality-based subscriptions but also supports subscriptions with range-matching. The range-matching is achieved by prepending the deterministically encrypted events with specially constructed prefixes. And again, the deterministic nature of the encryption leaves the protocol vulnerable to the statistical attack.

To overcome these deficiencies, [RR06] describes an approach that relies on searchable encryption schemes [EG07a; Goh03; SWP00]. The use of such encryption schemes allows for performing equality-based subscriptions for probabilistically encrypted event messages. Unfortunately, the highest level of security achieved by existing searchable encryption schemes is indistinguishability up to frequency (Definition 3.2.1): while performing a search for some value, a party that processes a search query infers that all found ciphertexts represent the same value. This still makes the statistical attack possible, though not so direct. The authors of the protocol admit this deficiency and argue that since subscriptions should always be somehow matched

against encrypted messages, such leaks are inevitable.

In [SL07], Srivatsa et al. propose avoiding this problem by routing event messages along different, probabilistically chosen, non-overlapping paths. In that way, they try to "smooth out" the event frequencies so that they will appear similar for every event. Such an approach can indeed help in making the observed distribution of the events less skewed. However, due to the limited number of non-overlapping paths in the broker network it cannot guarantee the confidentiality when a sufficient number of events is published. In addition, the authors rely on an entropy-based security model that does not provide formal evidence on the achieved level of the confidentiality.

5.1.3 Our Contribution

A number of works aimed at designing content-based routing systems that provide confidentiality of published events was published in recent years. However, as we have discussed in the previous section, none of them could present a complete solution to the problem. Moreover, in the all the proposed solutions the primary aim is to prevent non-trusted brokers from inferring details about routed events by concealing the execution of the routing procedure. However, none of them consider the attack in which a broker performs analysis of the incoming and outgoing messages. We take a closer look at such a kind of attack and show that it may enable the broker to infer sensitive information about published events, even if the routing procedure is completely obfuscated.

In this chapter, we try to answer the question as to whether it is possible to organize content-based routing in such a way that it is efficient and does not reveal anything about routed events to the non-trusted brokers. To do so, we (*i*) build a rigorous model that formally defines a content-based routing system that does not leak any information about routed events; and (*ii*) provide a formal proof that it is impossible to build an efficient content-based routing protocol that could preserve the confidentiality of the routed events and perform any better than a system that broadcasts events to all the subscribers. In our proof we consider the adversary that has a very limited view on the environment it operates in: we allow every routing algorithm to be running in a "black box" so that the adversaries cannot make any observations on its execution. Finally, we show that by slightly "loosening" the structure of the routing network and reducing the adversary's a priori knowledge about its topology, the confidentiality and the efficiency are still achievable.

5.2 Confidentiality in a Content-Based Routing System

In this section, we formalize the problem of secure content-based routing in a non-trusted environment. Furthermore, we discuss the possibility of constructing a content-based routing system that could provide both security and efficiency. After we ascertain that these requirements cannot be satisfied together, we propose a new architecture for a system that can perform content-based routing efficiently and ensure confidentiality of the routed messages.

5.2.1 Basic Idea

The primary aim of this chapter is to present a rigorous treatment to the notion of confidentiality of a content-based routing system in which brokers are not trusted. In order to do so, we first present a formal description of a class of systems that allow content-based routing. We start with introducing the notation and basic concepts. We then present a definition of a content-based routing system on which we can base a formal discussion of confidentiality aspects. When formulating the definition, our aim is to achieve as much generality as possible. In that way, we will be able to apply our results to the broad class of architectures that can be referred to as "content-based routing systems". Therefore, we will abstract from any implementation details referring to routing procedures as to algorithms of an arbitrary nature.

By using these definitions, we develop a formal security model that defines properties of a content-based routing system that guarantees confidentiality of published events when the routing network is not trusted: i.e., every broker is considered to be malicious. Assuming Kerckhoffs' principle, we consider an adversary that knows the topology of the network and the procedure, according to which the content-based routing system is constructed and deployed in the network.

Surprisingly, it appears that in such a model the ability to deliver events confidentially seems to be incompatible with an intuitive notion of efficiency of a content-based routing system. Even if we assume that the routing procedure is completely obfuscated and reveals absolutely no details about how and what routing decisions are made to the adversary, the confidentiality still remains unachievable.

Therefore, we relax our initial setting by assuming that the adversary has no initial knowledge of the network topology and cannot learn the topology while performing the routing. By presenting a content-based routing system

that is confidential and efficient in such relaxed model, we prove that such an assumption indeed helps us in achieving our goals.

5.2.2 Definition of a Content-Based Routing System

In this section, we present a formal description of a content-based routing system. Consider a network with topology defined by graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V} = \{P, b_1, \ldots, b_m, s_1, \ldots, s_l\}$ is a set of nodes and $\mathcal{E}$ is a set of edges connecting the nodes. Node P represents a publisher, nodes $b_1, \ldots, b_m$ represent m brokers and nodes $s_1, \ldots, s_l$ represent l subscribers. The nodes of the network can be connected in an arbitrary way and in the most general case all nodes are interconnected pairwise.

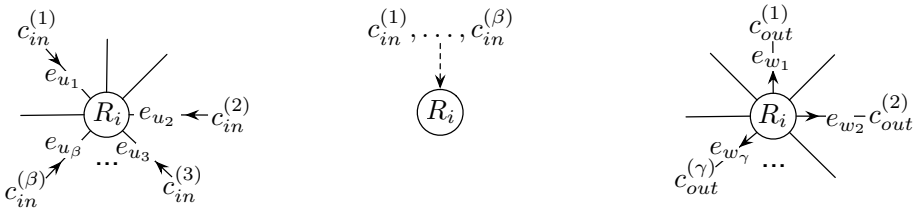
Usually it is agreed that an event consist of one or several pairs $\langle \alpha : \beta \rangle$ where α defines a subject and β defines its value (e.g. $\langle$temperature : 36$\rangle$, $\langle$pressure : 720$\rangle$). A subscriber can be interested either in all events containing specific subjects (e.g., it will be receiving all events containing information about the temperature) or it can additionally impose conditions on their values (e.g., it will be receiving events about the pressure only if its value exceeds 730). For now we will restrict ourselves to systems where only subject-bases subscriptions are possible.

The publisher publishes events with a subject belonging to subject space $\mathcal{X}$ and subscribers may subscribe for some of these subjects. That is, each subscriber s_i is assigned a *subscription* $\mathcal{S}_i \subset \mathcal{X}$. We say that a set of subscriptions $\{\mathcal{S}_1, \ldots, \mathcal{S}_l\}$ is *trivial* if all subscriptions are either empty or contain the whole subject space: $\mathcal{S}_i = \emptyset$, $i = 1, \ldots, l$ or $\mathcal{S}_i = \mathcal{X}$, $i = 1, \ldots, l$.

Before an event is published, the publisher transforms it into one or several *packets* that are transformed back to the original event by subscribers that receive all of these packets. It is obvious that if the aim is to preserve confidentiality of the events, the transformation will involve encrypting the event and thus, will require a key as one of the inputs.

Let E and D be transformation algorithms that map events to sequences of packets and $k \in \mathcal{K} = \{0,1\}^n$ is a key used by encryption and decryption procedures that are called as subroutines within the transformation algorithms: $E : \mathcal{K} \times \mathcal{X} \mapsto \mathcal{C}$, $D : \mathcal{K} \times \mathcal{C} \mapsto \mathcal{X}$ where $\mathcal{C}$ is a set of all possible packet sequences representing an event from $\mathcal{X}$. If there is no confidentiality requirement, no encryption is necessary and the key is simply ignored.

Usually it is assumed that for all the packets a point of entry to the broker network is always the same (i.e., the publisher always uses the same edge for communicating with the broker network). However, we allow the publisher to directly communicate with several nodes of the network (e.g., for the network displayed in Figure 5.3a publisher P can communicate with



(a) Incoming messages (b) Incoming messages with anonymous senders (c) Outgoing messages

Figure 5.2: Routing procedure

nodes b_1 and b_2).

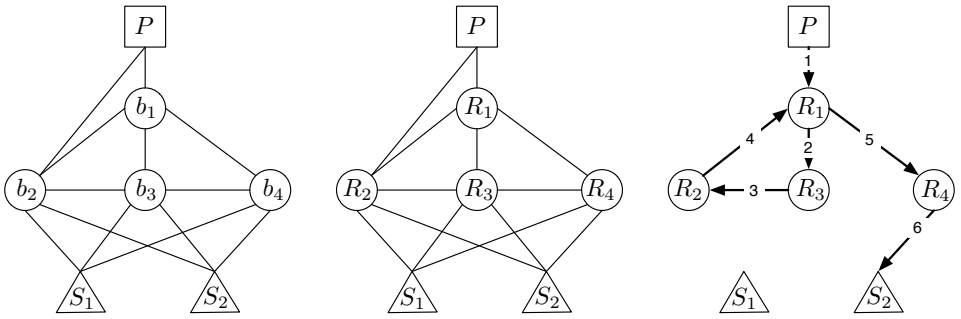
In the literature, a procedure performing the routing is usually described as a table of ⟨filter, edges⟩ tuples (in actual implementations instead of an edge a network address of a recipient is used). The table entries define along which edges a packet that satisfies a condition expressed by the filter should be forwarded. To generalize our result, we abstract from the actual routing mechanism and simply describe it as an *algorithm*. The routing algorithm R_i running on broker b_i which is adjacent to edges $e_1, \ldots, e_\alpha$ proceeds as follows:

1. Broker b_i receives incoming packets, each arriving along one of the adjacent edges as depicted on Figure 5.2a: $\text{In}_{b_i} = ((c_{\text{in}}^{(1)}, e_{u_1}), \ldots, (c_{\text{in}}^{(\beta)}, e_{u_\beta}))$, $\beta \leq \alpha$. Consider, that it also might be possible to receive packets without getting any knowledge about their sender when a packet does not contain the address of its source as depicted on Figure 5.2b: $\text{In}_{b_i} = ((c_{\text{in}}^{(1)}, e_{u_1}), \ldots, (c_{\text{in}}^{(\beta)}, e_{u_\beta}))$, $\beta \leq \alpha$.

2. R_i processes In_{b_i} and outputs outgoing packets with corresponding edges as depicted on Figure 5.2c: $R_i(\text{In}_{b_i}) = \text{Out}_{b_i}$,$\text{Out}_{b_i} = ((c_{\text{out}}^{(1)}, e_{w_1}))$, $\ldots, (c_{\text{out}}^{(\gamma)}, e_{w_\gamma})), \gamma \leq \alpha$. Note, that the outgoing packets are not necessarily the same as the incoming ones. This is because R_i can also apply some transformations to the input.

We define a single invocation of a routing algorithm (Figures 5.2b - 5.2c) as a *routing step*. We note that routing a single event may result in some brokers performing more than one routing step. For example, consider the route traversed by a packet on its way to subscriber s_2 as displayed in Figure 5.3c where broker b_1 performs two routing steps.

Traditionally, content-based routing is considered to be a deterministic operation: the probability that broker b_i forwards packet c to broker b_j is



(a) Physical network topology (b) A content-based routing network (c) A route of a packet in a content-based routing network

Figure 5.3: Content-based routing

either 0 or 1. We will be assuming that routing algorithms R_i can be probabilistic and stateful: the probability that broker b_i will route an incoming packet to b_j can be between 0 and 1, and also this probability can also depend on previous routing steps performed by R_i.

Now we are ready to formally describe a network that performs content-based routing.

Definition 5.2.1. *A* content-based routing network Λ *is a tuple* $(\mathcal{G}, R_1, \ldots, R_m, S_1, \ldots, S_l, k, E, D)$ *consisting of*

- *network topology* $\mathcal{G}$
- *probabilistic stateful routing algorithms* $R_1, \ldots, R_m$ *that are assigned to the corresponding brokers*
- l *subscriptions* $\{S_1, \ldots, S_l\}$, $S_i \subset \mathcal{X}$
- *transformation algorithms* E, D *with randomly and uniformly chosen key* $k \in \{0,1\}^n$ *known to the publisher and the subscribers.*

A content-based routing network Λ can be viewed as an overlay network that is built upon a physical network described by topology $\mathcal{G}$ (Figure 5.3a and 5.3b).

Since the purpose of such a network is to deliver published events to the corresponding subscribers within a reasonable amount of time, we introduce a notion of *correctness*:

Definition 5.2.2. *A content-based routing network* $\Lambda = (\mathcal{G}, R_1, \ldots, R_m, S_1, \ldots, S_l, k, E, D)$ *is* correct *if it satisfies the following criteria*

- *if $x \in \mathcal{S}_i$ then published event x always reaches subscriber s_i*

- *the routing of every event is performed in time bounded by some poly(n)*

Note, that this definition allows events to reach subscribers that are not subscribed for them (we also refer to them as to *false positives*). However, if the network allows such behavior, the subscribers are responsible for post-filtering the received events in order to sift out false positives.

A broker can observe a sequence of incoming and outgoing packets while an event is routed in the network. This can be defined as a broker's *view*.

Definition 5.2.3. *A* view *of broker b_i on the routing of event x is a sequence of accompanied by corresponding edges incoming and outgoing packets that are routed by b_i and that correspond to event x:*

$$\texttt{View}_{b_i}(x) = ((\mathrm{In}_b^{(1)}, \mathrm{Out}_b^{(1)}) \ldots, (\mathrm{In}_{b_i}^{(w)}, \mathrm{Out}_{b_i}^{(w)}))$$

If no packets are routed through broker b_i while routing event x, the corresponding view is considered to be empty: $\texttt{View}_{b_i}(x) = (\,)$.

Analogously, we define a view on the routing of a sequence of events.

Definition 5.2.4. *A* view *of broker b_i on the routing of sequence of events $\bar{x} = (x^{(1)}, \ldots, x^{(q)})$ is a sequence of accompanied by corresponding edges incoming and outgoing packets that are routed by b_i and that correspond to events from $\bar{x}$:*

$$\texttt{View}_{b_i}(\bar{x}) = ((\mathrm{In}_{b_i}^{(1)}, \mathrm{Out}_{b_i}^{(1)}) \ldots, (\mathrm{In}_{b_i}^{(w)}, \mathrm{Out}_{b_i}^{(w)}))$$

Note that we cannot describe such a view as a sequence of views on the routings of individual events of the sequence ($\texttt{View}_{b_i}(\bar{x}) = (\texttt{View}_{b_i}(x^{(1)}), \ldots, \texttt{View}_{b_i}(x^{(q)}))$) since a broker might not always be able to attribute pair $(\mathrm{In}_{b_i}^{(j)}, \mathrm{Out}_{b_i}^{(j)})$ to the corresponding event. It would be possible under the assumption that an event is published to the network only after all packets corresponding to an event published earlier have reached their destinations and are not present in the network anymore. However, this assumption does not hold in systems where intervals between published events are very short or the content-based routing system is not order preserving. In this case, the broker network is allowed to contain packets corresponding to various events and the order in which they arrive to the brokers may not correspond to the order in which they were published.

It is usually assumed in the literature that content-based routing systems provide a built-in mechanism for creating and updating routing tables

[MFP06]. Such a mechanism allows for dynamically adjusting routing algorithms in response to new subscriptions. However, since we are interested exclusively in the confidentiality of the events, we assume that the subscribers define their subscriptions before the events are published. Additionally, to simplify the setting, for now we assume that a trusted party generates and assigns the routing algorithms to the corresponding brokers.

Given network topology $\mathcal{G}$ and subscriptions $S_1, \ldots, S_l$ there might exist more than one way of organizing the routing. Thus, for fixed subscriptions there might exist a number of correct content-based routing networks that have the same topology but have different routing algorithms running on the brokers. To give a definition of such a general construction, we introduce a notion of a *content-based routing system.*

Definition 5.2.5. *A* content-based routing system *is a PPT algorithm* $\Re$ *that given key* $k \in \mathcal{K}$*, network topology* $\mathcal{G}$ *and subscriptions* $\{S_1, \ldots, S_l\}$*,* $S_i \subset \mathcal{X}$ *generates a correct content-based routing network:* $\Re : \mathcal{K} \times \mathcal{G} \times \underbrace{\mathcal{X} \times \ldots \times \mathcal{X}}_{l} \mapsto \Lambda$.

We do not put any implicit restriction on the amount of traffic transmitted through the network while routing the packets. The routing algorithm can be sending events along intricate paths, create "fake" packets and unnecessary cycles trying to prevent adversaries from making assumptions about events by analyzing routing patterns. The only restriction is that the amount of bytes transferred along any edge of the network should be bound by some polynomial, as it follows from the requirement to perform routing in polynomial time.

5.2.3 Confidentiality

We base our notion of events confidentiality in a content-based routing system with non-trusted broker network on the ideas captured in the definitions of indistinguishable security of encryption schemes presented in Section 2.5.

We assume that brokers, although being malicious, follow the routing protocol (such adversaries are often referred to as semi-honest) and do not collude with each other. We also traditionally assume that the only secret parameter of the system is the key, leaving the system implementation, network topology and subscriptions a common knowledge.

Given a network with topology $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V} = \{P, b_1, \ldots, b_m, s_1, \ldots, s_l\}$ and adversarial PPT algorithms $A_1, \ldots, A_m$, where A_i runs on broker b_i consider the following experiment performed for some fixed i:

1. The challenger randomly and uniformly chooses key k from key space: $k \xleftarrow{R} \mathcal{K}$.

2. The adversary chooses two sequences of events of the same length $\bar{x}_1 = (x^{(1)}, \ldots, x^{(q)}), \bar{x}_2 = (y^{(1)}, \ldots, y^{(q)})$, $q = q(n) \leq poly(n)$, chooses subscriptions $S_1, \ldots, S_l$, and gives it all to the challenger.

3. Using key k and subscriptions $S_1, \ldots, S_l$ the challenger uses content-based routing system $\Re$ to generate content-based routing network $\Lambda = (\mathcal{G}, R_1, \ldots, R_m, S_1, \ldots, S_l, k, E, D)$.

4. The challenger randomly and uniformly chooses $\beta \in \{1, 2\}$ and hands $\bar{x}_\beta$ to the publisher which publishes the sequence to the network .

5. Given topology $\mathcal{G}$, routing algorithm R_i and the view of broker b_i on the routing of $\bar{x}_\beta$ algorithm A_i tries to guess β.

6. If A_i guesses β correctly the outcome of the experiment is 1 and 0 otherwise.

A content-based routing system $\Re$ is called indistinguishably secure if the probability that any A_i can determine β correctly cannot be significantly higher then the probability of guessing it randomly.

Formally it can be described as follows:

Experiment $Exp(A_i, b_i)$
$k \xleftarrow{R} \mathcal{K} = \{0,1\}^n$
Let $\bar{x}_1 = (x^{(1)}, \ldots, x^{(q)}), \bar{x}_2 = (y^{(1)}, \ldots, y^{(q)})$,
$q = q(n) \leq poly(n)$
Let $S_1 \subset \mathcal{X}, \ldots, S_l \subset \mathcal{X}$
$\Lambda \leftarrow \Re(k, \mathcal{G}, \mathcal{S}_1, \ldots, \mathcal{S}_l)$
$\beta \xleftarrow{R} \{1, 2\}$
publish $\bar{x}_\beta$ to Λ
$g \leftarrow A_i(\mathcal{G}, R_i, \texttt{View}_{b_i}(\bar{x}_\beta))$
if $\beta = g$ **return** 1
else return 0

Definition 5.2.6. *Content-based routing system $\Re$ is indistinguishably secure if for every broker b_i, every PPT A_i, every positive polynomial $p(\cdot)$, every set of subscriptions $S_1, \ldots, S_l$, every pair of event sequences $\bar{x}_1 =$*

$(x^{(1)}, \ldots, x^{(q)}), \bar{x}_2 = (y^{(1)}, \ldots, y^{(q)})$, $q = q(n) \leq poly(n)$ *and all sufficiently large* n *the probability of* $Exp(A_i, b_i)$ *returning 1 cannot be significantly greater than the probability of guessing* β *by drawing it randomly and uniformly from set* $\{1, 2\}$*:*

$$\Pr[Exp(A_i, b_i) = 1] - \frac{1}{2} < \frac{1}{p(n)}, \text{ for all } i = 1, \ldots, m$$

As an informal argument behind this definition, consider a publisher that, while being in state **A**, periodically publishes event x and, while being in state **B**, periodically publishes event y. A network generated by an indistinguishably secure content-based routing system will not contain brokers that could notice that the publisher has transitioned from state **A** to state **B** by observing a change in the sequence of published events. In other words, for any subscriptions $S_1, \ldots, S_l$ none of the brokers should be able to distinguish between event sequences $(x, \ldots, x, \ldots)$ and $(x, \ldots, y, \ldots)$.

As an example of a content-based routing system that is indistinguishably secure consider a system that *broadcasts* encrypted events to all the subscribers along some fixed paths. If an indistinguishably secure (for multiple messages) encryption scheme is used for encrypting these events, then the views on routings of any sequences will also be indistinguishable from each other. Obviously, such a system satisfies Definition 5.2.6. But since in that case every published event will reach a subscriber independently of whether it was subscribed for it, such an approach can be considered extremely impractical.

5.2.4 Limitations for Confidential Content-Based Routing Systems

As we noted earlier, in some works it was mentioned that it might be impossible to build an efficient content-based routing system that could guarantee events' confidentiality when the brokers are not trusted. As a reason it was noted that in order to perform a routing, the brokers should be able to match encrypted events against corresponding subscription filters, thereby allowing gathering statistics on frequencies of the published events.

It is true that so far no technique was proposed that could allow performing a search on encrypted data and reveal nothing while the search is performed. However, it is not known whether building such an algorithm is an unsolvable problem. The same can be referred to the routing algorithms. In [BGI+01] it was shown that there is no general way to build an *obfuscator* that could conceal details of any algorithm. An obfuscator can be informally

described as a "compiler" that takes an algorithm as an input and outputs an obfuscated version of the same algorithm, hiding all details about its execution and achieving a "virtual black box" property. However, such a negative result does not rule out the possibility that *some* algorithms could be obfuscated. Therefore, until the contrary is proven, the possibility of being able to obfuscate a content-based routing algorithm cannot be excluded.

Also the following workaround may help in preventing statistical attacks based on the ability of the adversary to track the frequencies of searched values:

1. Each event x_i is assigned counter $q_i = 0$ that is incremented each time x_i is published.

2. A published event x_i is concatenated with the corresponding counter and deterministically encrypted as $E_k(x_i|q_i)$.

3. A broker responsible for routing this event contains a buffer of subscription filters for $E_k(x_1|q_1), \ldots, E_k(x_p|q_p)$, where p is a fixed parameter.

4. After the broker routes the event, the filter corresponding to the just routed $E_k(x_i|q_i)$ is discarded.

5. Periodically the publisher (or some other trusted entity) refills the buffers of the brokers with new filters.

With such an approach, each filter is used only once, thus allowing the filter matching without leaking information about frequencies of the events.

A more practical approach could be to perform content-based routing using a secure coprocessor. Then each broker with a routing algorithm running within such a coprocessor could match probabilistically encrypted packets against corresponding filters without revealing any information on how this matching is done. Alternatively, if the routing table cannot fit in RAM of the coprocessor, the coprocessor can implement a PIR protocol that will also prevent the adversary from seeing which of the $\langle filter, edge \rangle$ tuples of the routing table was used.

But, as we show now, there is another obstacle that renders the building of a confidential and, at the same time, efficient content-based routing system an impossible task.

Theorem 9. *There is no indistinguishably secure correct content-based routing system that allows non-trivial subscriptions and has average amounts of data post-filtered by the subscribers significantly lower than in a system that performs routing by broadcasting all events to the subscribers.*

First we prove the following lemmas:

Lemma 6. *Content-based routing system $\Re$ is indistinguishably secure if for every broker b_i, every PPT algorithm A_i, every positive polynomial $p(\cdot)$, every set of subscriptions $\{S_1, \ldots, S_l\}$, every pair of event sequences $\bar{x}_1 = (x^{(1)}, \ldots, x^{(q)})$, $\bar{x}_2 = (y^{(1)}, \ldots, y^{(q)})$, $q = q(n) \leq \text{poly}(n)$ and all sufficiently large n,*

$$|\Pr[\mathbf{Exp}(A_i, b_i) = 1] - \Pr[\mathbf{Exp}(A_i, b_i) = 0]| < \frac{1}{p(n)}$$

Proof. According to Definition 5.2.6, if content-based routing system $\Re$ is indistinguishably secure, then for every broker b_i, every PPT algorithm A_i, every positive polynomial $p'(\cdot)$ and all sufficiently large n,

$$\Pr[\mathbf{Exp}(A_i, b_i) = 1] - \frac{1}{2} < \frac{1}{p'(n)}$$

From the existence of PPT algorithm A_i that while running on broker b_i guesses β with probability that cannot be significantly greater than 1/2 follows the existence of PPT algorithm $\hat{A}_i$ that uses A_i is a subroutine and outputs 1 (0) when A outputs 0 (1). Clearly, the probability of $\hat{A}_i$ guessing β also cannot be significantly smaller than 1/2. Thus, the inequality from Definition 5.2.6 can be rewritten as

$$|\Pr[Exp(A_i, b_i) = 1] - \frac{1}{2}| < \frac{1}{p'(n)}$$

Considering that $\Pr[Exp(A_i, b_i) = 0] = 1 - \Pr[Exp(A_i, b_i) = 1]$ we can also write it as

$$|\Pr[Exp(A_i, b_i) = 0] - \frac{1}{2}| < \frac{1}{p'(n)}$$

Summing up two last inequalities and recalling that $|a + b| \leq |a| + |b|$ we get

$$|\Pr[Exp(A_i, b_i) = 1] - \Pr[Exp(A_i, b_i) = 0]| < \frac{2}{p'(n)} = \frac{1}{p(n)}$$

□

Lemma 7. *For every broker b_i and every edge e_j adjacent to it, for every $N \in \mathbb{N}$, every positive polynomial $p(\cdot)$, every set of subscriptions $\{S_1, \ldots, S_l\}$, every pair of event sequences $\bar{x}_1 = (x^{(1)}, \ldots, x^{(q)}), \bar{x}_2 = (y^{(1)}, \ldots, y^{(q)})$, $q = q(n) \leq \text{poly}(n)$ and all sufficiently large n,*

$$|\Pr[N^{out}_{b_i,e_j} = N|\beta = 1] - \Pr[N^{out}_{b_i,e_j} = N|\beta = 2]| < \frac{1}{p(n)}$$

where $N^{out}_{b_i,e_j}$ *is the number of bits transferred by broker* b_i *along edge* e_j *while sending outgoing packets to its neighbors and* $\Pr[N^{out}_{b_i,e_j} = N|\beta = 1]$, ($\Pr[N^{out}_{b_i,e_j} = N|\beta = 2]$) *is the conditional probability of the event in which the total number of bits contained in outgoing packets transferred by broker* b_i *along edge* e_j *under the condition that sequence* $\bar{x}_1$ ($\bar{x}_2$) *was chosen is equal to* N*.*

Proof. Assume that the condition of the lemma does not hold true and that there exists broker b_i and adjacent to it edge e_j for which there exist positive polynomial $p(\cdot)$, event sequences $\bar{x}_1 = (x^{(1)}, \ldots, x^{(q)}), \bar{x}_2 = (y^{(1)}, \ldots, y^{(q)})$, $q = q(n) \leq \text{poly}(n)$ and such N that there always exists sufficiently large n for which

$$|\Pr[N^{out}_{b_i,e_j} = N|\beta = 1] - \Pr[N^{out}_{b_i,e_j} = N|\beta = 2]| \geq \frac{1}{p(n)}$$

We show now that it allow to build a distinguisher that can distinguish between sequences $\bar{x}_1$ and $\bar{x}_2$ with significant probability.

Without losing generality we assume that

$$\Pr[N^{out}_{b_i,e_j} = N|\beta = 1] - \Pr[N^{out}_{b_i,e_j} = N|\beta =2] \geq \frac{1}{p(n)}$$

We construct distinguisher A_i that gets the view of broker b_i as its input and outputs 1 if $N^{out}_{b_i,e_j} = N$, otherwise it tries to guess β by randomly and uniformly drawing it from set $\{1, 2\}$:

$A_i(\texttt{View}_{b_i}(\bar{x}_\beta))$
1 : Using $\texttt{View}_{b_i}(\bar{x}_\beta)$ compute $N^{out}_{b_i,e_j}$
2 : **if** $N^{out}_{b_i,e_j} = N$ **return** 1
3 : $g \xleftarrow{R} \{1, 2\}$
4 : **return** g

We show now that if such A_i is used as a distinguisher in experiment $\mathbf{Exp}(A_i, b_i)$, then difference $\Pr[\mathbf{Exp}(A_i, b_i) = 1] - \Pr[\mathbf{Exp}(A_i, b_i) = 0]$ will be significant. Using the law of total probability we can rewrite $\Pr[\mathbf{Exp}(A_i, b_i) = 1]$ as

$$\begin{aligned}\Pr[\mathbf{Exp}(A_i, b_i) = 1] = \\ \Pr[\mathbf{Exp}(A_i, b_i) = 1|\beta = 1]\Pr[\beta = 1]+ \\ \Pr[\mathbf{Exp}(A_i, b_i) = 1|\beta = 2]\Pr[\beta = 2]\end{aligned}$$

Considering that $\Pr[\beta = 1] = \Pr[\beta = 2] = 1/2$ and that $\mathbf{Exp}(A_i, b_i)$ outputs 1 when $A_i(\mathtt{View}_{b_i}(\bar{x}_\beta))$ guesses β correctly, and 0 otherwise, we can further rewrite it as

$$\begin{aligned}\Pr[\mathbf{Exp}(A_i, b_i) = 1] = \\ \frac{1}{2}\Pr[A_i(\mathtt{View}_{b_i}(\bar{x}_\beta)) = 1|\beta = 1]+ \\ \frac{1}{2}\Pr[A_i(\mathtt{View}_{b_i}(\bar{x}_\beta)) = 2|\beta = 2]\end{aligned}$$

Applying the law of total probability to the definition of conditional probability we get

$$\begin{aligned}\Pr[A|B] = &\frac{\Pr[A \cap B]}{\Pr[B]} = \frac{\sum_i \Pr[A \cap B \cap C_i]}{\Pr[B]} = \\ &\frac{\sum_i \Pr[A|B \cap C_i]\Pr[B \cap C_i]}{\Pr[B]} = \\ &\sum_i \Pr[A|B \cap C_1]\Pr[C_i|B]\end{aligned}$$

where $\{C_i : i = 1, 2, \ldots\}$ is a countable finite partitioning of a probability space.

That allows us to rewrite $\Pr[A_i(\mathtt{View}_{b_i}(\bar{x}_\beta)) = 1|\beta = 1]$ and $\Pr[A_i(\mathtt{View}_{b_i}(\bar{x}_\beta) = 2|\beta = 2]$ as

$$\begin{aligned}&\Pr[A_i(\mathtt{View}_{b_i}(\bar{x}_\beta)) = 1|\beta = 1] = \\ &\sum_i \Pr[A_i(\mathtt{View}_{b_i}(\bar{x}_\beta)) = 1|\beta = 1 \cap N^{out}_{b_i,e_j} = i] \times \Pr[N^{out}_{b_i,e_j} = i|\beta = 1]\end{aligned} \tag{5.1}$$

$$\begin{aligned}&\Pr[A_i(\mathtt{View}_{b_i}(\bar{x}_\beta)) = 2|\beta = 2] = \\ &\sum_i \Pr[A_i(\mathtt{View}_{b_i}(\bar{x}_\beta)) = 2|\beta = 2 \cap N^{out}_{b_i,e_j} = i] \times \Pr[N^{out}_{b_i,e_j} = i|\beta = 2]\end{aligned} \tag{5.2}$$

Recall, that in all cases other then when $N^{out}_{b_i,e_j} = N$ algorithm A_i randomly chooses its output from $\{1, 2\}$ and, therefore,

$$\begin{aligned}&\Pr[A_i(\mathtt{View}_{b_i}(\bar{x}_\beta)) = 1|\beta = 1 \cap N^{out}_{b_i,e_j} = i] = \frac{1}{2},\ i \neq N, \\ &\Pr[A_i(\mathtt{View}_{b_i}(\bar{x}_\beta)) = 1|\beta = 1 \cap N^{out}_{b_i,e_j} = N] = 1, \\ &\Pr[A_i(\mathtt{View}_{b_i}(\bar{x}_\beta)) = 2|\beta = 2 \cap N^{out}_{b_i,e_j} = i] = \frac{1}{2},\ i \neq N, \\ &\Pr[A_i(\mathtt{View}_{b_i}(\bar{x}_\beta)) = 2|\beta = 2 \cap N^{out}_{b_i,e_j} = N] = 0.\end{aligned}$$

Thus, (5.1) and (5.2) can be rewritten as

$$\Pr[A_i(\mathtt{View}_{b_i}(\bar{x}_\beta)) = 1|\beta = 1] = \tag{5.3}$$
$$\sum_{i \neq N} \frac{1}{2}\Pr[N^{out}_{b_i,e_j} = i|\beta = 1] + \Pr[N^{out}_{b_i,e_j} = N|\beta = 1]$$

$$\Pr[A_i(\mathtt{View}_{b_i}(\bar{x}_\beta)) = 2|\beta = 2] = \tag{5.4}$$
$$\sum_{i \neq N} \frac{1}{2}\Pr[N^{out}_{b_i,e_j} = i|\beta = 2] + \Pr[N^{out}_{b_i,e_j} = N|\beta = 2],$$

respectively.

Repeating the analogous transformations for $\Pr[\mathbf{Exp}(A_i, b_i) = 0]$ we get

$$\begin{aligned}
&\Pr[\mathrm{Exp}(A_i, b_i) = 0] = \\
&\Pr[\mathrm{Exp}(A_i, b_i) = 0|\beta = 1]\mathbf{Pr}[\beta = 1] + \\
&\Pr[\mathbf{Exp}(A_i, b_i) = 0|\beta = 2]\Pr[\beta = 2] = \\
&\frac{1}{2}\Pr[A_i(\mathtt{View}_{b_i}(\bar{x}_\beta)) = 2|\beta = 1] + \\
&\frac{1}{2}\Pr[A_i(\mathtt{View}_{b_i}(\bar{x}_\beta)) = 1|\beta = 2]
\end{aligned}$$

$$\begin{aligned}
&\Pr[A_i(\mathtt{View}_{b_i}(\bar{x}_\beta)) = 2|\beta = 1] = \qquad (5.5)\\
&\sum_i \Pr[A_i(\mathtt{View}_{b_i}(\bar{x}_\beta)) = 2|\beta = 1 \cap N^{out}_{b_i,e_j} = i]\Pr[N^{out}_{b_i,e_j} = i|\beta = 1] = \\
&\sum_{i \neq N} \frac{1}{2}\Pr[N^{out}_{b_i,e_j} = i|\beta = 1]
\end{aligned}$$

$$\begin{aligned}
&\Pr[A_i(\mathtt{View}(b_i(\bar{x}_\beta)) = 1|\beta = 2] = \qquad (5.6)\\
&\sum_i \Pr[A_i(\mathtt{View}_{b_i}(\bar{x}_\beta)) = 1|\beta = 2 \cap N^{out}_{b_i,e_j} = i]\Pr[N^{out}_{b_i,e_j} = i|\beta = 2] = \\
&\sum_{i \neq N} \frac{1}{2}\Pr[N^{out}_{b_i,e_j} = i|\beta = 2] + \Pr[N^{out}_{b_i,e_j} = N|\beta = 2]
\end{aligned}$$

Summing the right parts of (5.3) and (5.4) and subtracting the right parts

of (5.5) and (5.6) we get

$$\begin{aligned}
&\Pr[\mathbf{Exp}(A_i, b_i) = 1] - \Pr[\mathbf{Exp}(A_i, b_i) = 0] = \\
&\sum_{i \neq N} \frac{1}{2} \Pr[N^{out}_{b_i,e_j} = i | \beta = 1] + \Pr[N^{out}_{b_i,e_j} = N | \beta = 1] + \\
&\sum_{i \neq N} \frac{1}{2} \Pr[N^{out}_{b_i,e_j} = i | \beta = 2] - \sum_{i \neq N} \frac{1}{2} \Pr[N^{out}_{b_i,e_j} = i | \beta = 1] - \\
&\sum_{i \neq N} \frac{1}{2} \Pr[N^{out}_{b_i,e_j} = i | \beta = 2] - \Pr[N^{out}_{b_i,e_j} = N | \beta = 2] = \\
&\Pr[N^{out}_{b_i,e_j} = N | \beta = 1] - \Pr[N^{out}_{b_i,e_j} = N | \beta = 2] \geq \frac{1}{p(n)}
\end{aligned}$$

This contradicts Lemma 6 concluding the proof. □

Using the result obtained in Lemma 7 we can now show that it is impossible to construct a content-based routing system that could ensure confidentiality and be on average more efficient than when all events are simply broadcasted to the subscribers, thus proving Theorem 9.

Proof. Consider content-based routing system $\Re$ with topology $\mathcal{V} = \{P, b_1, \ldots, b_m, s_1, \ldots, s_l\}$ and event space $\mathcal{X}$. We fix some non-trivial set of subscriptions $\{S_1, \ldots, S_l\}$ and from this set choose subscription S_{i_0} that is a proper subset of event space $\mathcal{X}$.

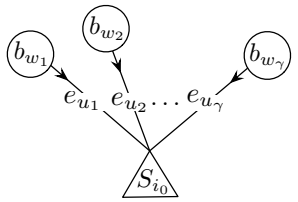


Figure 5.4: Fragment of a network topology

Let $\bar{x}_1 = (x^{(1)}, \ldots, x^{(q)}), q = q(n) \leq poly(n), x^{(i)} \in S_{i_0}$ be a sequence consisting of events that are members of subscription S_{i_0} and $\bar{x}_2 = (y^{(1)}, \ldots, y^{(q)})$, $y^{(i)} \in \mathcal{X}$ be a sequence consisting of arbitrary events belonging to $\mathcal{X}$.

Let $b_{w_1}, \ldots, b_{w_\gamma}$ be brokers adjacent to s_{i_0} and connected to it with edges $e_{u_1}, \ldots, e_{u_\gamma}$ as displayed on Figure 5.4. Since we allow routing algorithms to be probabilistic, the amounts of bits transferred along edges $e_{u_1}, \ldots, e_{u_\gamma}$ can be seen as random variables with corresponding probability mass functions

$\Pr_{\bar{x}_\beta}[N_{e_{u_1}} = N], \ldots, \Pr_{\bar{x}_\beta}[N_{e_{u_\gamma}} = N]$, where $\Pr_{\bar{x}_\beta}[N_{e_{u_i}} = N]$ is the probability that N bits are routed along edge e_{u_i} when sequence $\bar{x}_\beta$ is published.

In Lemma 7 it was shown that for every N, every positive polynomial $p(\cdot)$ and all sufficiently large n,

$$|\Pr_{\bar{x}_1}[N_{e_{u_i}} = N] - \Pr_{\bar{x}_2}[N_{e_{u_i}} = N]| < \frac{1}{p(n)}$$

Multiplying left and right parts of the inequality by N we get

$$|E_{\bar{x}_1}(N_{e_{u_i}}(N)) - E_{\bar{x}_2}(N_{e_{u_i}}(N))| < \frac{N}{p(n)}$$

where $E_{\bar{x}_1}(N_{e_{u_i}}(N))$ and $E_{\bar{x}_2}(N_{e_{u_i}}(N))$ are expected values for the amounts of bits routed along edge e_{u_i} when $\bar{x}_1$ and $\bar{x}_2$ are routed correspondingly.

Summing over i and applying $|a + b| \leq |a| + |b|$ rule we get

$$|\sum_{i=1}^{\gamma} E_{\bar{x}_1}(N_{e_{u_i}}(N)) - \sum_{i=1}^{\gamma} E_{\bar{x}_2}(N_{e_{u_i}}(N))| < \frac{Nu_\gamma}{p(n)}$$

Since the sum of expected values of random variables (dependent or independent) is equal to the expected value of the sum of random variables it can be rewritten as

$$|E_{\bar{x}_1}(\sum_{i=1}^{\gamma} N_{e_{u_i}}(N)) - E_{\bar{x}_2}(\sum_{i=1}^{\gamma} N_{e_{u_i}}(N))| < \frac{Nu_\gamma}{p(n)}$$

Recall, that according to Definition 5.2.2 a routing should be performed in polynomial time. Therefore, there exists a polynomial $p'(\cdot)$ that bounds the amount of routed along any of the edges bits N, and the inequality can be rewritten as

$$|E_{\bar{x}_1}(\sum_{i=1}^{\gamma} N_{e_{u_i}}(N)) - E_{\bar{x}_2}(\sum_{i=1}^{\gamma} N_{e_{u_i}}(N))| < \frac{u_\gamma}{\frac{p(n)}{p'(n)}}$$

which is a negligible value.

The latter can be interpreted as a difference of numbers of bits that on average reach S_{i_0} when sequences $\bar{x}_1$ and $\bar{x}_2$ correspondingly are routed in the network. Since all events from sequence $\bar{x}_1$ are members of S_{i_0} and $\bar{x}_2$ can be composed of arbitrary events, it follows that no matter what events are published, the average amount of data that should be post-filtered by subscriber s_{i_0} differs negligibly from the average amount of post-filtered data in case the events were broadcasted and all of them were delivered to s_{i_0}. Since s_{i_0} can represent any subscriber of the system that concludes the proof of the theorem. □

The proof can be trivially extended to topologies that have more than one publisher. This is because multiple publishers can be considered to be a single one that publishes all the events and is connected to the same brokers.

Note that the edge along which the routing algorithm decides to forward an outgoing packet cannot be hidden from the adversary since in order to perform the routing the broker should be able to read the recipient's address. Thus, the adversarial algorithm constructed in Lemma 7 can always be used by a malicious broker.

Another important observation is that distinguisher A_i does not require any knowledge about functioning of routing algorithm R_i for guessing which sequence was routed in the network. This means that even if the routing algorithm could be completely obfuscated and the adversary could merely observe incoming and outgoing packets, a system that should perform better than the system where the events are broadcasted still could not provide complete confidentiality of the routed events.

Finally, the assumption we made about the trusted party used for constructing the routing algorithms and delivering them to the corresponding brokers in no way affects the result of Theorem 9. If such a party does not exist, and, as it is usually assumed in the literature, the brokers participate in the construction and distribution of the routing algorithms, it can only increase adversaries' knowledge about the system, and hence, increases their chances in guessing the content of the routed messages.

5.2.5 Relaxed Model

In the previous section, we have shown that if a network topology is known to adversaries, confidentiality is only achievable when all the performance gain of an intelligent routing system is sacrificed. In this section, we consider a relaxed model in which the adversaries have no knowledge of the network topology. We may allow them to know the number of brokers, subscribers and publishers, but they otherwise have no further knowledge of the network. The wording of the relaxed definition of the indistinguishable security of a content-based routing system is almost identical to Definition 5.2.6 with the only difference in the description of experiment $\mathbf{Exp}(A_i, b_i)$ that does not have topology $\mathcal{G}$ as one of the inputs of algorithm A_i.

However, it may be possible for a broker to start obtaining information about the topology by analyzing the flow of the packets. To avoid it, the system should not allow the broker to learn its "neighborhood" by examining destinations of the packets it forwards to adjacent nodes. When sending

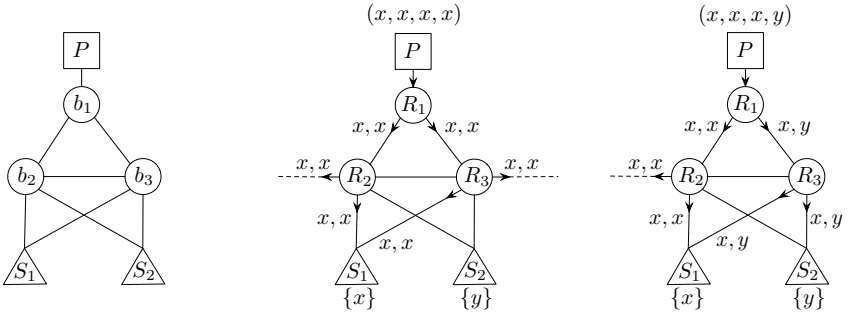
packets as datagrams[1], it is possible to organize communications in such a way that a broker will not be able to determine whether packets it sends are addressed to an existing or to a "faked" neighbor. The usage of datagrams is realistic if it can be assumed that either the network connection is 100% reliable or the loss of some packets is acceptable. We also assume that the routing is performed in a "black box" by using one of the approaches mentioned in Section 5.2.4. Under these assumptions, we can show that it is possible to have a content-based routing system that satisfies the relaxed version of Definition 5.2.6 and is more efficient than the broadcast (in terms of the computation overhead of the subscribers).

We prove the existence of such a system by the following example. Consider event space $\mathcal{X} = \{x, y\}$ and a network displayed in Figures 5.5a. Assume that every event has the same length and is indistinguishably encrypted, the brokers have no knowledge of the network topology and that a broker has no way of knowing whether a recipient of the packet it sends physically exists. Under thess assumptions, we illustrate a way to securely route some sequences of event with a very modest number of false positives. The goal is to make all views of the brokers on routings of all possible event sequences of the same length indistinguishable by always sending the same numbers of encrypted packets to the same numbers of recipients (some of which may be not existing). An example of such routing for subscriptions $S_1 = \{x\}, S_2 = \{y\}$ is depicted in Figures 5.5b - 5.5d (routings of sequences (x, y, y, y) and (y, y, y, y) are symmetric to the routings of sequences (x, x, x, y) and (x, x, x, x)) and for subscriptions $S_1 = \{x, y\}, S_2 = \{y\}$ is depicted on Figures 5.5e - 5.5i. It is trivial to construct routing patterns for remaining subscription sets ($\{S_1 = \{x\}, S_2 = \{y, y\}\}, \{S_1 = \{y\}, S_2 = \{x\}\}$, etc.). A dotted line means that the packets are forwarded to the non-existing ("faked") recipient. Note that for bigger networks there can be more than one dotted line adjacent to a broker. This means that the packets are sent to several faked recipients.

The indistinguishable security of such routing trivially follows from the following observation: for every sequence the brokers behavior is indistinguishable since they always receive the same amount of indistinguishably encrypted packets and always route the same numbers of packets along the same numbers of edges. Moreover, routing patterns depicted on Figures 5.5b, 5.5d, 5.5e, 5.5g, 5.5i have no false positives at all, delivering to subscribers only those events they are subscribed to.

Clearly, when the event space is bigger and the topology is more labyrin-

[1]A datagram is a block of data that is transferred across the network and does not require that the recipient send a confirmation to the sender acknowledging its receipt.



(a) Network topology

(b) Routing of (x, x, x, x), $S_1 = \{x\}$, $S_2 = \{y\}$

(c) Routing of (x, x, x, y), $S_1 = \{x\}$, $S_2 = \{y\}$

(d) Routing of (x, x, y, y), $S_1 = \{x\}$, $S_2 = \{y\}$

(e) Routing of (x, x, x, x), $S_1 = \{x, y\}$, $S_2 = \{y\}$

(f) Routing of (x, x, x, y), $S_1 = \{x, y\}$, $S_2 = \{y\}$

(g) Routing of (x, x, y, y), $S_1 = \{x, y\}$, $S_2 = \{y\}$

(h) Routing of (x, x, y, y), $S_1 = \{x, y\}$, $S_2 = \{y\}$

(i) Routing of (x, x, y, y), $S_1 = \{x, y\}$, $S_2 = \{y\}$

Figure 5.5: Example of indistinguishably secure routing system

thine, the number of false positives may increase. However, if taken over many different subscriptions and event sequences, on average such an approach will noticeably outperform broadcasting in terms of the numbers of

the messages that will have to be examined by the subscribers in order to sift out false positives.

The provided example illustrates that under certain conditions content-based routing *can* be performed confidentially and efficiently. Elaborating on the details of such a routing system is outside the scope of this thesis. Proposing ways to efficiently construct and deploy such a routing system, ensuring that its computation and communicative overheads are suitable for practical applications is a topic for future research.

5.3 Summary

Having a content-based routing system that is able to preserve confidentiality of the routed messages can greatly benefit the applicability of such an architecture by reducing security risks that are inevitable when sensitive information is disseminated across multiple parties. However, to implement such a system, trade-offs between efficiency and security are necessary. In this chapter, we have shown that the provable confidentiality is achievable only if severe limitations are imposed on the efficiency of the routing. We believe that our result offers new insights on potential vulnerabilities and threats for content-based routing systems.

We have also defined a relaxed model that limits the adversaries' knowledge about the network topology and illustrated that under such assumptions it might be possible to have a content-based routing system that guarantees confidentiality of the events that are routed in a non-trusted environment and performs noticeably better than the broadcasting approach. Constructing routing algorithms that could efficiently provide confidential routing in such a model, minimizing the number of false positives is an interesting topic for further research.

Chapter 6

Conclusion

In this thesis, we analyzed a problem of secure outsourcing of IT-services to an application service provider that is not trusted. Very often the fact that the service will be provided by some third party, which requires entrusting this party with sensitive service-related data, and the fear that it can opportunistically and unnoticeably use the obtained data outbalances the benefits of the outsourcing. That gives rise to the following question: Is it possible to outsource a service without revealing a content of the data which are required by the service?

We tried to answer this question regarding services related to the outsourcing of private databases and of content-based routing of sensitive data. In doing so, we were adhering to notions of security generally accepted in the cryptographic community. In many respects, such an approach deprived us of flexibility in constructing schemes that could arguably propose some level of security. Yet, it allowed us to rely on rigorous approaches when describing characteristics of our solutions, presenting formal proofs of security based on sound theoretical foundations.

Below we briefly summarize the key contributions of this thesis:

1. **Secure database outsourcing.** We provided a formalization of the problem of secure database outsourcing, examined its application limits and proposed several solutions that fit into the proposed security model.

   (a) **Security definitions.** We proposed a definition that formally captures the intuition behind the notion of secure database outsourcing. Relying on various examples, we illustrated that our definition indeed defines a system that conceals all information about data contained in the outsourced database and that the definition is not too strong in the sense that any relaxation will cause a security breach in the system. We also defined necessary conditions

for the definition which allowed us to estimate complexity bounds imposed on architectures supporting secure database outsourcing. Unfortunately, it appeared that these bounds fix computation and communicative overheads on a level that rules out any practical solution as the trivial approach where the client downloads the whole database and processes the data herself seems to be more efficient. Taking this into account, we proposed a relaxed version of the security definition where we allowed the service provider to infer information about the number of tuples affected by an exact select query, but nothing more. Furthermore, we outlined requirements for a system that could be considered secure in such a relaxed model and described practical scenarios to which such a model can be applied.

(b) **Framework for secure database outsourcing.** Exploiting a similarity between full-text search and certain database operations, we defined a framework that allows us to reuse encryption schemes that enable search on encrypted data for constructing secure database outsourcing solutions. Giving a formal treatment to the notion of "similarity," we were able to describe security of the framework using the security definitions we proposed earlier. Using an exemplary searchable encryption scheme, we described a secure database outsourcing solution that supports a practically relevant query language that includes exact select, exact delete and insert operations and allowing to use AND and OR logical operations in `WHERE` conditions of the corresponding SQL queries. The resulting solution, although supporting a limited number of database operations, offers much better security, if compared to the existing works.

(c) **Searchable encryption scheme.** We constructed a searchable encryption scheme with characteristics superior to the existing counterparts and provided a formal proof of its security. Compared to the analogous searchable encryption schemes, all of which with high probability allow false positives in the resulting set of a search operation, our scheme guarantees that the probability of such errors is negligible. When applied to the proposed framework for secure database outsourcing, the scheme allows for increasing the number of feasible database operations: it additionally allows for performing projection and exact update and, in addition to AND and OR, use NOT logical operation in `WHERE` conditions of the corresponding SQL queries.

(d) **Discretionary access control.** Considering as a motivating example a scenario in which participants of an RFID-enabled supply chain are willing to share product-related data, we proposed a technique that allows for defining discretionary read/write access rules for such data. An example can be generalized as an outsourced database that stores data that are provided and accessed by multiple clients and gives the clients the ability to define access rules for data they provide without any involvement of the service provider. Compared to existing approaches, our technique does not require permissions of the users to form a hierarchy and efficiently supports dynamic changes of permissions for new or existing users. We conducted a series of performance tests and evaluated their results against a supply chain of a large (existing) company, confirming the practicability of the proposed approach.

2. **Secure outsourcing of content-based routing operations.** We considered a problem of the possibility of performing a content-based routing that preserves confidentiality of routed messages in a non-trusted network.

   (a) **Confidentiality of content-based routing.** We developed a definition that, to our knowledge, is the first to provide a formal treatment for confidentiality in content-based routing systems. The definition is applicable to a broad class of systems and, using notions generally accepted in the cryptographic community, formally defines an adversarial model and captures the requirement to reveal nothing about routed messages to an adversary.

   (b) **Relationship between confidentiality and efficiency.** Examining the possibility of constructing a routing protocol that could satisfy the proposed definition of confidentiality and an intuitive notion of efficiency, we established that confidentiality and efficiency cannot be satisfied simultaneously. Therefore, we provided a relaxed version of the confidentiality definition in which we considered a content-based routing system with an architecture that is slightly different from that generally accepted in literature and in a realistic way limits knowledge of the adversary about the system. By building a content-based routing system that satisfied the relaxed definition we confirmed its relevance and practicability.

In conclusion, the problem of secure outsourcing of IT services still may be considered open. Tight security requirements on the one hand, and the

necessity to provide reach functionality and efficiency on the other render finding a solution that could satisfy them both extremely difficult. Moreover, as we have shown, for some operations the notions of security and efficiency appear to be contradictory. How can a lower complexity be achieved for scenarios and operations that are shown to be feasible? Which other operations cannot be securely and efficiently outsourced? We hope that we helped in answering some of these questions. But still much remains to be done before secure outsourcing becomes a generally accepted concept, removing a barrier of trust between service providers and their potential customers.

Bibliography

[ABCS06] Anderson, Ross; Bond, Mike; Clulow, Jolyon; Skorobogatov, Sergei: Cryptographic processors – a survey. In: *Proceedings of the IEEE*, volume 94(2):pp. 357–369, 2006.

[AF02] Asonov, Dmitri; Freytag, Johann Christoph: Almost optimal private information retrieval. In: *Privacy Enhancing Technologies*, pp. 209–223. 2002.

[Ama08] Amazon.com: Privacy notice. URL f http://www.amazon.com/gp/help/customer/display.html?nodeId=468496#sharel, 2008.

[AT83] Akl, Selim G.; Taylor, Peter D.: Cryptographic solution to a problem of access control in a hierarchy. In: *ACM Trans. Comput. Syst.*, volume 1(3):pp. 239–248, 1983. ISSN 0734-2071.

[BBB+07] Barker, Elaine; Barker, William; Burr, William; Polk, William; Smid, Miles: Recommendation for key management. NIST special publication 800-57, revised, National Institute of Standards and Technology (NIST), May 2007.

[BCOP04] Boneh, Dan; Crescenzo, Giovanni; Ostrovsky, Rafail; Persiano, Giuseppe: Public-key Encryption with Keyword Search. In: *Proceedings of Eurocrypt*, volume 3027 of *Lecture Notes in Computer Science*. 2004.

[Bds02] German Federal Data Protection Act (Bundesdatenschutzgesetz), 2002. URL `www.bdd.de/Download/bdsg_eng.pdf`.

[BG02] Boyens, Claus; Günther, Oliver: Trust Is not Enough: Privacy and Security in ASP and Web Service Environments. In: *Sixth East-European Conference on Advances in Databases and Information Systems*, volume 2435 of *Lecture Notes In Computer Science*. 2002.

[BG03] Boyens, Claus; Günther, Oliver: Using online services in untrusted environments: a privacy-preserving architecture. In: *ECIS*. 2003.

[BGI+01] Barak, Boaz; Goldreich, Oded; Impagliazzo, Russell; Rudich, Steven; Sahai, Amit; Vadhan, Salil P.; Yang, Ke: On the (im)possibility of obfuscating programs. In: *CRYPTO '01: Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology*, pp. 1–18. Springer-Verlag, London, UK, 2001. ISBN 3-540-42456-3.

[BKOW07] Boneh, Dan; Kushilevitz, Eyal; Ostrovsky, Rafail; William E. Skeith III: Public Key Encryption that Allows PIR Queries. Cryptology ePrint Archive, Report 2007/073, 2007. `http://eprint.iacr.org/2007/073/`.

[BL76] Bell, David; LaPadula, Leonard: Secure computer systems: Unified exposition and multics interpretation. In: *MITRE technical report, MITRE Corporation, Bedford Massachusetts*, volume 2997:p. ref A023 588, 1976.

[Ble98] Bleichenbacher, Daniel: Chosen Ciphertext Attacks Against Protocols Based on The RSA Encryption Standard PKCS#1. In: *Advances in Cryptology*. 1998.

[Boy04] Boyens, Claus: *Privacy Trade-offs in Web-Based Services*. Ph.D. thesis, Humboldt-Universität zu Berlin, 2004.

[BS05] Bertino, Fellow-Elisa; Sandhu, Fellow-Ravi: Database security-concepts, approaches, and challenges. In: *IEEE Trans. Dependable Secur. Comput.*, volume 2(1):pp. 2–19, 2005. ISSN 1545-5971.

[CBB03] Cilia, Mariano; Bornhövd, Christof; Buchmann, Alejandro P.: Cream: An infrastructure for distributed, heterogeneous event-based applications. In: *CoopIS/DOA/ODBASE*, pp. 482–502. 2003.

[CG97] Chor, Benny; Gilboa, Niv: Computationally private information retrieval (extended abstract). In: *STOC '97: Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pp. 304–313. ACM, New York, NY, USA, 1997. ISBN 0-89791-888-6.

[CGKS95] Chor, Benny; Goldreich, Oded; Kushilevitz, Eyal; Sudan, Madhu: Private Information Retrieval. In: *IEEE Symposium on Foundations of Computer Science*. 1995.

[CM05] Chang, Yan-Cheng; Mitzenmacher, Michael: Privacy preserving keyword searches on remote encrypted data. In: *ACNS*, pp. 442–455. 2005.

[CMS99] Cachin, Christian; Micali, Silvio; Stadler, Markus: Computationally private information retrieval with polylogarithmic communication. In: *Lecture Notes in Computer Science*, volume 1592:pp. 402–414, 1999.

[CS98] Cramer, Ronald; Shoup, Victor: A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In: *CRYPTO*, pp. 13–25. 1998.

[DdVF+05] Damiani, Ernesto; di Vimercati, S. De Capitani; Foresti, Sara; Jajodia, Sushil; Paraboschi, Stefano; Samarati, Pierangela: Key management for multi-user encrypted databases. In: *StorageSS '05: Proceedings of the 2005 ACM workshop on Storage security and survivability*, pp. 74–83. ACM Press, New York, NY, USA, 2005.

[DFHJ98] Domingo-Ferrer, Josep; Herrera-Joancomartí, Jordi: A Privacy Homomorphism Allowing Field Operations on Encrypted Data, 1998.

[Dis00] Disabatino, Jennifer: Disney offers to buy toysmart.com customer list. CNN News Online, URL `http://archives.cnn.com/2000/TECH/computing/07/14/disney.toysmart.list.idg`, Jule 2000.

[DMS04] Dingledine, Roger; Mathewson, Nick; Syverson, Paul: Tor: The second-generation onion router. In: *Proceedings of the 13th USENIX Security Symposium*. 2004.

[DVJ+03] Damiani, Ernesto; Vimercati, S. De Capitani; Jajodia, Sushil; Paraboschi, Stefano; Samarati, Pierangela: Balancing Confidentiality and Efficiency in Untrusted Relational DBMSs. In: *CCS '03: Proceedings of the 10th ACM Conference on Computer and Communications Security*. ACM Press, 2003. ISBN 1-58113-738-9.

[EFG06] Evdokimov, Sergei; Fischmann, Matthias; Günther, Oliver: Provable security for outsourcing database operations. In: *ICDE '06: Proceedings of the 22nd International Conference on Data Engineering (ICDE'06)*. IEEE Computer Society, 2006.

[EG07a] Evdokimov, Sergei; Günther, Oliver: Encryption techniques for secure database outsourcing. In: *ESORICS*, pp. 327–342. 2007.

[EG07b] Evdokimov, Sergei; Günther, Oliver: Practical access control management for outsourced epc-related data in rfid-enabled supply chain. In: *ICEBE '07: Proceedings of the IEEE International Conference on e-Business Engineering*, pp. 331–336. IEEE Computer Society, Washington, DC, USA, 2007. ISBN 0-7695-3003-6.

[EPC06] EPCglobal: EPCglobal Tag Data Standards Version 1.3, March 2006.

[FMG02] Fiege, Ludger; Mühl, Gero; Gärtner, Felix C.: Modular event-based systems. In: *The Knowledge Engineering Review*, volume 17(4):pp. 359–388, 2002. ISSN 0269-8889.

[Gam84] Gamal, Taher El: A public key cryptosystem and a signature scheme based on discrete logarithms. In: *CRYPTO*, pp. 10–18. 1984.

[GM84] Goldwasser, Shafi; Micali, Silvio: Probabilistic encryption. In: *J. Comput. System Sci.*, volume 28(2):pp. 270–299, 1984. ISSN 0022-0000.

[Goh03] Goh, Eu-Jin: Secure indexes. Cryptology ePrint Archive: Report 2003/216, 2003. URL `http://eprint.iacr.org/2003/216/`.

[Gol01] Goldreich, Oded: *Foundations of Cryptography*, volume Basic Tools. Cambridge University Press, 2001.

[Gol04] Goldreich, Oded: *Foundations of Cryptography*, volume Basic Applications. Cambridge University Press, 2004.

[HILM02] Hacıgümüş, Hakan; Iyer, Bala; Li, Chen; Mehrotra, Sharad: Executing SQL over Encrypted Data in the Database-Service-Provider Model. In: *Proceedings of the 28th SIGMOD Conference on the Management of Data*. ACM, 2002.

[HL90] Harn, Lein; Lin, Hung-Yu: A cryptographic key generation scheme for multilevel data security. In: *Comput. Secur.*, volume 9(6):pp. 539–546, 1990. ISSN 0167-4048.

[HY03] Hwang, Min-Shiang; Yang, Wei-Pang: Controlling access in large partially ordered hierarchies using cryptographic keys. In: *J. Syst. Softw.*, volume 67(2):pp. 99–107, 2003. ISSN 0164-1212.

[IS05] Iliev, Alexander; Smith, Sean W.: Protecting client privacy with trusted computing at the server. In: *IEEE Security and Privacy*, volume 3(2):pp. 20–28, 2005. ISSN 1540-7993.

[KBC97] Krawczyk, Hugo; Bellare, Mihir; Canetti, Ran: HMAC: Keyed-Hashing for Message Authentication, 1997.

[Ker83] Kerckhoffs, Auguste: La cryptographie militaire. In: *Journal des sceiences militaires*, volume IX:pp. 5–38, jan 1883.

[KO97] Kushilevitz, Eyal; Ostrovsky, Rafail: Replication is not needed: single database, computationally-private information retrieval. In: *FOCS '97: Proceedings of the 38th Annual Symposium on Foundations of Computer Science (FOCS '97)*, p. 364. IEEE Computer Society, Washington, DC, USA, 1997. ISBN 0-8186-8197-7.

[LLS04] Li, Jun; Lu, Chenghuai; Shi, Weidong: An efficient scheme for preserving confidentiality in content-based publish-subscribe systems. Technical report, Georgia Institute of Technology, 2004.

[MFP06] Mühl, Gero; Fiege, Ludger; Pietzuch, Peter: *Distributed Event-Based Systems*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006. ISBN 3540326510.

[MTMA85] MacKinnon, Stephen J.; Taylor, Peter D.; Meijer, Henk; Akl, Selim G.: An optimal algorithm for assigning cryptographic keys to control access in a hierarchy. In: *IEEE Trans. Comput.*, volume 34(9):pp. 797–802, 1985. ISSN 0018-9340.

[Nat77] National Institute of Standards, U.S. Department of Commerce: *FIPS 47: Data encryption standard*, January 1977.

[Nat01] National Institute of Standards, U.S. Department of Commerce: *FIPS 197: Advanced encryption standard*, November 2001.

[OP01] Opyrchal, Lukasz; Prakash, Atul: Secure distribution of events in content-based publish subscribe systems. In: *SSYM'01: Proceedings of the 10th conference on USENIX Security Symposium*, pp. 21–21. USENIX Association, Berkeley, CA, USA, 2001.

[RAD78] Rivest, Ronald L.; Adleman, Leonard; Dertouzos, Michael L.: On Data Banks and Privacy Homomorphisms. In: DeMillo, R.; Dobkin, D.; Jones, A.; Lipton, R., editors, *Foundations of Secure Computation*. Academic Press, 1978.

[RG02] Ramakrishnan, Raghu; Gehrke, Johannes: *Database Management Systems*. McGraw-Hill Science/Engineering/Math, 2002. ISBN 0072465638.

[RM04] Römer, Kay; Mattern, Friedemann: Event-based systems for detecting real-world states with sensor networks: A critical analysis. In: *DEST Workshop on Signal Processing in Sensor Networks at ISSNIP*, pp. 389–395. Melbourne, Australia, December 2004.

[RR06] Raiciu, Costin; Rosenblum, David S.: Enabling confidentiality in content-based publish/subscribe infrastructures. In: *Proc. Second IEEE Communications Society/CreateNet Int. Conf. on Security and Privacy in Communication Networks (SecureComm 2006)*. Aug.-Sep. 2006.

[RSA78] Rivest, Ronald L.; Shamir, Adi; Adleman, Leonard M.: A method for obtaining digital signatures and public-key cryptosystems. In: *Commun. ACM*, volume 21(2):pp. 120–126, 1978.

[San00] Sandoval, Greg: Failed dot-coms may be selling your private information. CNET News Archive, June 2000. URL `http://yahoo.cnet.com/news/0-1007-200-2176430.html`.

[SC07] Sion, Radu; Carbunar, Bogdan: On the computational practicality of private information retrieval. In: *NDSS '07: Proceedings of the Network and Distributed Systems Security Symposium (NDSS) '07*. 2007.

[Sha49] Shannon, Claude E.: Communication theory of secrecy systems. In: *Bell System Technical Journal*, volume 28:pp. 656–715, 1949.

[SKW+98] Schneier, Bruce; Kelsey, John; Whiting, Doug; Wagner, David; Hall, Chris: Twofish: A 128-bit block cipher, 1998.

[SL05] Srivatsa, Mudhakar; Liu, Ling: Securing publish-subscribe overlay services with eventguard. In: *CCS '05: Proceedings of the 12th ACM conference on Computer and communications security*, pp. 289–298. ACM, New York, NY, USA, 2005. ISBN 1-59593-226-7.

[SL07] Srivatsa, Mudhakar; Liu, Ling: Secure event dissemination in publish-subscribe networks. In: *ICDCS '07: Proceedings of the 27th International Conference on Distributed Computing Systems*, p. 22. IEEE Computer Society, Washington, DC, USA, 2007. ISBN 0-7695-2837-3.

[SW74] Stonebraker, Michael; Wong, Eugene: Access control in a relational data base management system by query modification. In: *ACM 74: Proceedings of the 1974 annual conference*, pp. 180–186. ACM Press, New York, NY, USA, 1974.

[SWP00] Song, Dawn Xiaodong; Wagner, David; Perrig, Adrian: Practical techniques for searches on encrypted data. In: *IEEE Symposium on Security and Privacy*. 2000.

[Tan06] Tang, Yi: Sharing session keys in encrypted databases. In: *ICEBE '06: Proceedings of the IEEE International Conference on e-Business Engineering*, pp. 47–54. IEEE Computer Society, Washington, DC, USA, 2006. ISBN 0-7695-2645-4.

[WCEW02] Wang, Chenxi; Carzaniga, Antonio; Evans, David; Wolf, Alexander L.: Security issues and requirements for internet-scale publish-subscribe systems. In: *Proceedings of the 35th Annual Hawaii International Conference on System Sciences (HICSS'02)-Volume 9*, p. 303. IEEE Computer Society, Washington, DC, USA, 2002. ISBN 0-7695-1435-9.

[WDDB06] Wang, Shuhong; Ding, Xuhua; Deng, Robert H.; Bao, Feng: Private information retrieval using trusted hardware. In: *ESORICS*, pp. 49–64. 2006.

[YZW06] Yang, Zhiqiang; Zhong, Sheng; Wright, Rebecca N.: Privacy-Preserving Queries on Encrypted Data. In: *Proceedings of the 11th European Symposium On Research In Computer Security (Esorics)*. 2006.

[Zet04] Zetter, Kim: Free e-mail with a steep price? Wired News URL http://www.wired.com/techbiz/media/news/2004/04/62917, April 2004.

Acknowledgement

This work would not have been possible without guidance and support of my doctoral advisor Oliver Günther. His insightful comments and advices inspired and directed my work throughout the time I spent at the Institute for Information Systems.

I am also grateful to Bharat Bhargava, my second advisor, who gave me an opportunity to visit Purdue University and working with whom was always a pleasure.

I thank the professors of the Berlin-Brandenburg Graduate School of Distributed Information Systems for their valuable feedback and motivation, especially Johann Christoph Freytag and Hans-Joachim Lenz.

I would also like to thank the fellow students of the graduate school and the university colleagues and friends who were a source of many new ideas, fun and inspiration. I am particularly grateful to my office mate Matthias Fischmann who played a role of my mentor in the early days of my work, Benjamin Fabian, and Holger Ziekow for productive and successful research cooperation, Lenka Ivantysynova, Christoph Goebel, Christoph Tribowski, and Runli Xie for a great working atmosphere.

Finally, I want to thank my family and my girlfriend Hanna for their invaluable support.

Selbständigkeitserklärung

Hiermit erkläre ich, die vorliegende Arbeit selbständing ohne fremde Hilfe verfasst und nur die angegebene Literatur und die angegebenen Hilfsmittel verwendet zu haben.